KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

Vaidotas Valiukas

APTARNAVIMO SISTEMŲ MODELIŲ TYRIMAS DEKOMPOZICIJOS METODU

Magistro darbas

Vadovas
doc. dr. E. Valakevičius

KAUNAS, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

TVIRTINU
Katedros vedėjas
prof. habil.dr. V. Pekarskas
2009 06 08

APTARNAVIMO SISTEMŲ MODELIŲ TYRIMAS DEKOMPOZICIJOS METODU

Matematikos magistro baigiamasis darbas

Vadovas
..... **doc. dr. E. Valakevičius**
2009 06 02

Recenzentas
..... **doc. dr. V. Pilkauskas**
2009 06 03

Atliko
FMMM 7 gr. stud.
..... **V. Valiukas**
2009 06 02

KAUNAS, 2009

KVALIFIKACINĖ KOMISIJA

Pirmininkas – Leonas Saulis, profesorius (VGTU).

Sekretorius – Eimutis Valakevičius, docentas (KTU).

Nariai: Algimantas Jonas Aksomaitis, profesorius (KTU),

Arūnas Barauskas, Vice-prezidentas projektams (UAB „Baltic Amadeus“),

Vytautas Janilionis, docentas (KTU),

Zenonas Navickas, profesorius (KTU),

Vidmantas Povilas Pekarskas, profesorius (KTU),

Rimantas Rudzkis, valdybos pirmininko patarėjas („DnB NORD” bankas).

Valiukas V. The analysis of queueing systems models by the decompositional method: Master's work in applied mathematics / supervisor dr. assoc. prof. E. Valakevičius; Department of mathematical research in systems, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2009. – 95 p.

SUMMARY

The approximation of general distributions by the mixtures of exponential distributions allows extending a class of queueing systems represented by Markovian processes. A decomposition of the transition matrix in order to find stationary probabilities of Markov chain is described in this paper. When the model is too large to analyze in its entirety, it is divided into subsystems. Each subsystem is analyzed separately and global solution is constructed from the partial solutions.

In order to compute the stationary probabilities the software was created. The analysis of *M/G/1* and *G/M/1* queueing systems showed that both of the *IAD* algorithms – *KMS* and *Takahashi* – are really effective to find stationary probabilities. A small modification or complement was introduced to the scheme of decomposition approach. Furthermore, the optimal strategy of the decomposition of *G/M/1* models was found.

SANTRAUKA

Realių aptarnavimo sistemų aproksimavimas Markovo procesais, naudojant eksponentinių skirstinių sąsūkas ar mišinius, apsunkina pačios sistemos analizę. Šiame darbe nagrinėjamos Markovo grandinės apytikslių stacionarių tikimybių radimo galimybės, taikant dekompozicijos principą. Nagrinėjant didelės apimties modelius, jie skaidomi į submodelius. Kiekvienas modelis analizuojamas atskirai ir bendras sprendinys suformuojamas iš dalinių sprendinių.

Sukurta programinė įranga aptarnavimo sistemų modelių *M/G/1* ir *G/M/1* stacionarių tikimybių skaičiavimui. Joje buvo realizuoti *KMS* ir *Takahashi* algoritmai. Minėtų aptarnavimo sistemų analizė parodė, jog dekompozicijos metodas efektyviai skaičiuoja stacionarias tikimybes. Tyrimų eigoje, atsižvelgiant į analizuojamų modelių subtilybes, buvo pasiūlytas būdas lengvesnei blokinės struktūros paieškai organizuoti. Tai leido *G/M/1* modelius visada suskaidyti į submodelius. Tuo tarpu *M/G/1* modelių tyrimas parodė, kad perėjimo tikimybių matricos skaidymas (atsižvelgiant į teorijos reikalavimus) į submatricas yra negalimas.

Darbo tematika pristatyta mokslinėje konferencijoje „Matematika ir matematikos dėstymas – 2009“.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Markovo procesai yra vieni nuodugniausiai ištyrinėtų ir plačiausiai praktikoje taikomų atsitiktinių procesų. Jų taikymo sritys – aptarnavimo teorija, informacinės sistemos, ekonomika ir kitos. Markovo procesus patogu taikyti įvairių sistemų modeliavimui ne tik dėl jų natūralaus panašumo į realias sistemas (kai sistema aprašoma tam tikra būsenų aibe, su perėjimais tarp jų), bet ir dėl nuodugniai išplėtotos matematinės teorijos.

Tolydaus laiko Markovo procesai glaudžiai susiję su eksponentiniu skirstiniu, nes perėjimo trukmė iš vienos būsenos į kitą pasiskirsčiusi pagal eksponentinį dėsnį. Realiose sistemose ši sąlyga dažnai nebūna išpildyta, todėl tai apsunkina jų tyrimą ir modeliavimą. Tačiau bendrojo tipo (tolygaus, gama ir kitų) skirstinių aproksimavimas eksponentinių skirstinių sąsūkomis ar mišiniais leidžia praplėsti stochastinių sistemų, kurios gali būti aprašomos Markovo procesais, klasę.

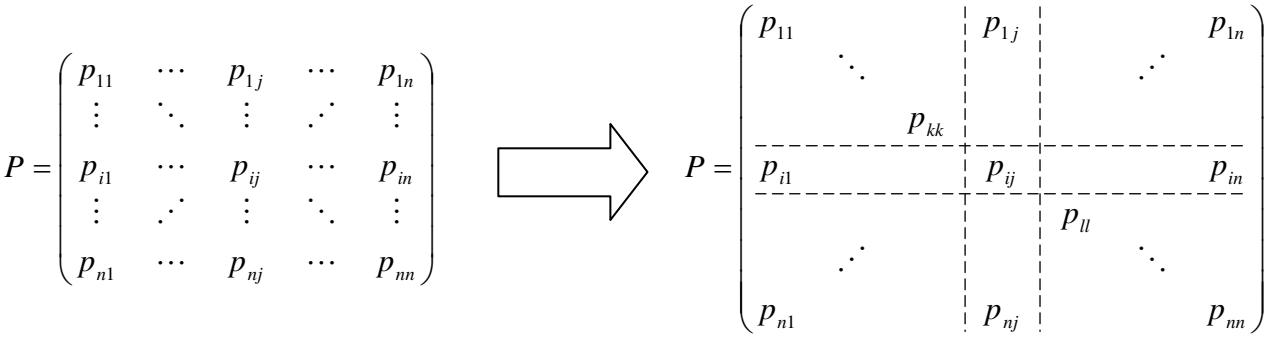
Vienas iš stochastinės sistemos analizės uždavinių yra tikimybinių charakteristikų apskaičiavimas. Tokiu atveju yra būtinos sistemos būsenų stacionarios tikimybės. Modeliuojant sudėtingas (aproksimuojant bendruosius skirstinius eksponentiniais mišiniais) sistemas, gauname didelio matavimo perėjimo tikimybių iš vienos būsenos į kitą matricą

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1j} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ p_{i1} & \cdots & p_{ij} & \cdots & p_{in} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nj} & \cdots & p_{nn} \end{pmatrix}.$$

Norint apskaičiuoti stacionarių tikimybių vektorių π, reikia spręsti lygčių sistemą

$$\pi \cdot P = \pi .$$

Kadangi klasikiniai lygčių sprendimo metodai dažnai neveikia dėl didelės skaičiavimo apimties bei paklaidų kaupimosi, todėl šiame darbe buvo panaudotas matricos dekompozicijos metodas – matricos skaidymo į smulkesnes matricas būdas, kai gautos dalinės sistemos analizuojamos atskirai ir po to suformuojamas bendras sistemos sprendinys.



pav. Dekompozicijos principo esmė

Jeigu apytiksles stacionarias tikimybes skaičiuojame naudodami dekompozicijos metodą, automatiškai iškyla dvi problemos:

- Kokiu būdu matrica P suskaidoma į mažesnes matricas;
- Kaip sudaromas bendras sprendinys π iš atskirų sprendinių.

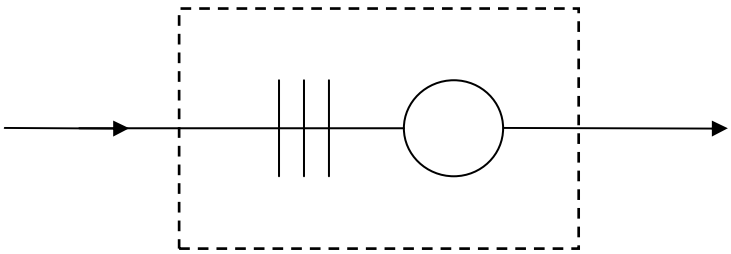
Pirmuoju atveju į pagalbą pasitelkėme grafų teorijos žinias – matrica P traktuojama kaip orientuoto grafo vaizdavimas. Tuomet ieškome stipriai sujungtų grafo viršūnių (plačiau apie grafus [2]). Antroji problema darbe sprendžiama realizuojant *KMS (Koury, McAllister, Stewart)* ir *Takahashi* algoritmus. Išsamus algoritmų aprašymas ir su jais susiję klausimai išnagrinėti literatūroje [9].

Skaitiniam aptarnavimo sistemų *M/G/1* ir *G/M/1* modeliavimui, kai *G* aproksimuojamas Kokso skirstiniu ar Erlango mišiniu, yra sukurta programinė įranga [10]. Papildžius turimą kompiuterinę sistemą dekompozicijos principą įgyvendinančiais metodais, šiame darbe tyrėme minėtas sistemas. Gauti rezultatai parodė, jog dekompozicijos metodas ne visais atvejais tinka skaičiuoti apytiksles sistemos stacionarias tikimybes.

1. TEORINĖ DALIS

1.1. APTARNAVIMO SISTEMOS MODELIS

Apibendrintas aptarnavimo sistemos modelis, kuris gali būti naudojamas operatorių užsakymų, ryšių įrangos informacijos apdorojimo, transporto srautų modeliavimui, pavaizduotas 1.1 paveiksle.



1.1 pav. Pagrindinis aptarnavimo sistemos modelis

Be kita ko, pagrindiniai aptarnavimo sistemas apibūdinantys komponentai [1,6]:

- **Atvykstančių paraiškų srautas.** Paprastai laikoma, kad visų paraiškų atvykimo laikai yra nepriklausomi ir vienodai pasiskirstę. Praktikoje dažniausiai naudojamas ir geriausiai ištirtas Puasoninis srautas. Puasoninio srauto laiko tarpai tarp paraiškų atvykimo pasiskirstę pagal eksponentinį dėsnį.
- **Aptarnavimo laikas.** Paprastai laikoma, kad kiekvienos paraiškos aptarnavimo laikas yra nepriklausomi vienodai pasiskirstę atsitiktiniai dydžiai.
- **Aptarnavimo pajėgumai.** T.y., aptarnavimo įrenginių skaičius.
- **Eilės dydis.** Gali būti baigtinis natūralusis skaičius arba begalybė – t.y., eilės ilgis neribotas. (Nors teoriniuose modeliuose daugelis formulių išvesta begalinėms eilėms, praktiniame sistemų modeliavime įvedami apribojimai.)
- **Aptarnavimo tvarka.** Paraiškos gali būti aptarnaujamos įvairia tvarka, pvz. *FIFO* (first in first out) – pirmiausia aptarnaujama seniausiai atėjusi paraiška. Taip pat gali egzistuoti prioritetai (vieno tipo paraiškos aptarnaujamos pirmiau negu kito tipo), arba įvairiai nustatoma tvarka esant daugiau nei vienam aptarnavimo įrenginiui.

Anglų statistiko M. Kendalo įvesta klasifikacija [1] leidžia sutrumpintai aprašyti aptarnavimo sistemą. Pagal šią notaciją sistema apibūdinama simbolių trejetu a/b/c. Pirma raidė žymi ateinančio srauto skirstinį, antra – aptarnavimo laiko skirstinį. Pavyzdžiui, raidė *M* žymi eksponentinį skirstinį (nuo angliško žodžio *memoryless*), *G* – bendrojo tipo skirstinį (vėlgi nuo angliško žodžio *general*), *D* – determinuotą, t.y. be jokio atsitiktinumo, apibrėžtą. Trečia raidė nurodo aptarnavimo įrenginių skaičių. Taigi, *M/G/1* žymi aptarnavimo sistemą, kurios paraiškų atėjimo srautas Puasoninis (paraiškų atvykimo laikai pasiskirstę pagal eksponentinį skirstinį), aptarnavimo laikas pasiskirstęs pagal skirstinį

G, o sistema turi vieną aptarnavimo įrenginį. Toks žymėjimas gali būti išplėstas pridedant daugiau raidžių. Pavyzdžiui, jei sistemos *M/G/1* maksimalus leidžiamas eilės ilgis *N*, tai rašome *M/G/1/N*.

Paprastai laikoma, kad paraiškos ateina po vieną, eilės ilgis neribojamas, o paraiškos aptarnaujamos atėjimo tvarka. Jei specialiai nenurodysime kitaip, toliau laikysimės tokių pačių prielaidų.

1.2. APTARNAVIMO SISTEMŲ SKAITMENINIS MODELIAVIMAS

Sistemoms modeliuoti naudosime kompiuterinę sistemą (Pranevičius, 1996), kuri leidžia analizuoti Markovo procesus ir yra ypač patogi modeliuoti aptarnavimo sistemas. Kompiuterinė sistema naudoja tam tikrą įvykių kalbą t.y., detaliai aprašomas kiekvienas įvykis, pakeičiantis sistemos būseną, bei būsenų apribojimai.

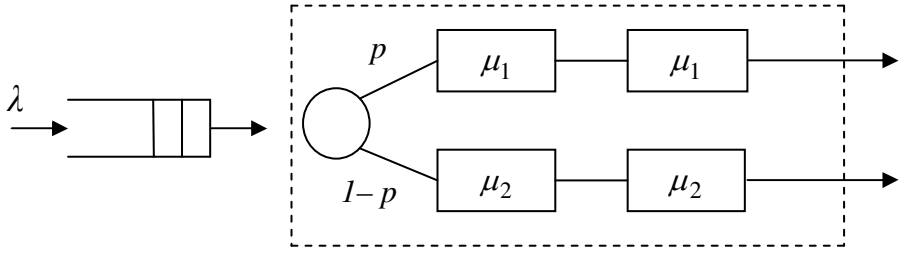
Panagrinėkime sistemą *M/M/1/N*, t.y. aptarnavimo sistemą su vienu aptarnavimo įrenginiu ir ribotu paraiškų skaičiumi *N*. Jeigu nagrinėjamas sistemos parametras x – paraiškų skaičius sistemoje, tai sistemą *M/M/1/N* galima interpretuoti kaip tolydaus laiko Markovo procesą su diskrečia būsenų aibe. Akivaizdu, kad tokio proceso būsenų skaičius yra $N + 1$. Tokio proceso stacionarių tikimybių apskaičiavimas, net ir esant nelabai dideliems *N* (pvz., 20) yra komplikuotas uždavinys.

Magistro baigiamajame darbe (Šnipas, 2008) buvo tiriamas stochastinių sistemų aproksimavimas Markovo modeliais. Aproksimavimui naudojamas fiktyvių fazių metodas – būsena, kurios skirstinys neeksponentinis (*G*), pakeičiama eksponentinių fazių mišiniu. Be to, minėta kompiuterinė sistema buvo pritaikyta aptarnavimo sistemų *M/G/1* ir *G/M/1* modeliavimui, atliekant bendrojo skirstinio *G* aproksimavimą Erlango mišiniais ir Kokso skirstiniu.

1.2.1. M/G/1 SISTEMŲ MODELIAVIMAS

1.2.1.1. M/G/1 SISTEMOS MODELIS, KAI G – ERLANGO MIŠINYS

Nagrinėjama sistema *M/G/1*, kai bendrojo tipo skirstinio funkcija *G* aproksimuojama antrojo tipo antros eilės Erlango mišiniu [10]. Aproksimuojamos sistemos fazinė schema:



1.2 pav. Sistemos M/G/1 aproksimavimo schema, kai G – Erlango mišinys

Sistema formaliai aprašoma įvykių kalba. Įvykių, kurie gali įvykti sistemoje, aibė yra $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$. Sistemos būsenų aibė: $N = \{(n_1, n_2)\}$, $n_1 = \overline{0, L}$, $n_2 = \overline{0,5}$.

e_1 – paraiška atėjo į aptarnavimo bloką, pirmąją (fiktyvią) fazę su intensyvumu λ;

e_2 – paraiška perėjo į aptarnavimo bloko antrą fazę su tikimybe p;

e_3 – paraiška perėjo į aptarnavimo bloko trečią fazę su tikimybe $1 - p$;

e_4 – paraiška perėjo į aptarnavimo bloko ketvirtą fazę su intensyvumu μ_1;

e_5 – paraiška perėjo į aptarnavimo bloko penktą būseną su intensyvumu μ_2;

e_6 – paraiška baigta aptarnauti ketvirtoje fazėje ir palieka sistemą su intensyvumu μ_1;

e_7 – paraiška baigta aptarnauti penktoje fazėje ir palieka sistemą su intensyvumu μ_2;

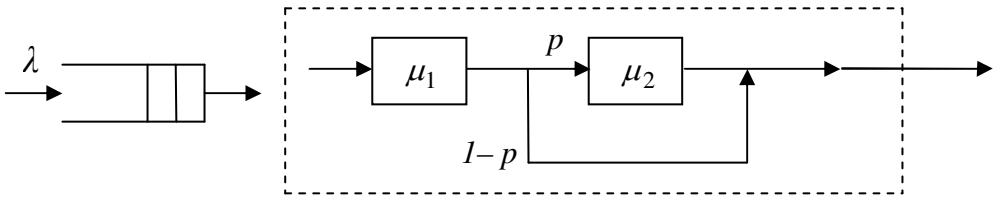
n_1 – paraiškų skaičius sistemoje (0 – jei sistema tuščia) ;

n_2 – parodo, kurioje aptarnavimo bloko fazėje aptarnaujama paraiška (0 – jei aptarnavimo blokas tuščias).

Paraiškų skaičius sistemoje n_1 apribotas tam tikru natūraliuoju skaičiumi L. Nors modeliuojame sistemą su begaline eile, praktikoje būtina įvesti apribojimą. Kitaip sakant, sistema *M/G/1/∞* aproksimuojama *M/G/1/L*. Reikiamas tikslumas pasiekiamas parinkus pakankamai didelį L.

1.2.1.2. M/G/1 SISTEMOS MODELIS, KAI G – KOKSO SKIRSTINYS

Nagrinėjama sistema *M/G/1*, kai bendrojo tipo skirstinio funkcija *G* aproksimuojama Kokso skirstiniu. Aproksimuojamos sistemos fazinė schema:



1.3 pav. Sistemos M/G/1 aproksimavimo schema, kai G – Kokso skirstinys

Tuo pačiu būdu kaip ir prieš tai buvusiame skyrelyje sistema formaliai aprašoma įvykių kalba. Įvykių, kurie gali įvykti sistemoje, aibė yra $E = \{e_1, e_2, e_3, e_4\}$. Sistemos būsenų aibė: $N = \{(n_1, n_2)\}$, $n_1 = \overline{0, L}$, $n_2 = \overline{0,2}$ [10].

e_1 – paraiška atėjo į aptarnavimo bloką, pirmąją fazę su intensyvumu λ;

e_2 – paraiška perėjo į aptarnavimo bloko antrą fazę su tikimybe p ir intensyvumu μ_1;

e_3 – paraiška baigta aptarnauti pirmoje fazėje ir palieka sistemą su tikimybe $1-p$ ir intensyvumu μ_1;

e_4 – paraiška baigta aptarnauti antroje fazėje ir palieka sistemą su intensyvumu μ_2;

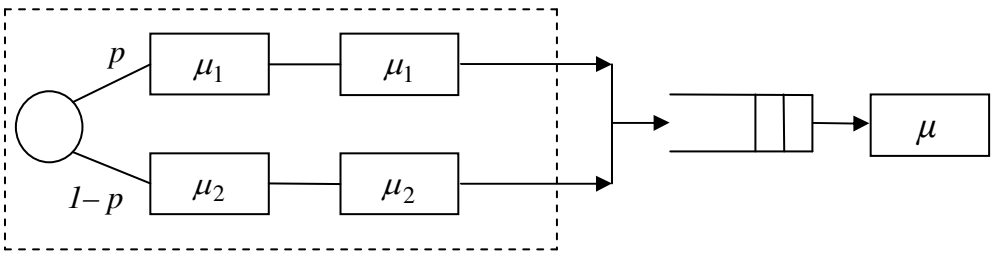
n_1 – paraiškų skaičius sistemoje (0 – jei sistema tuščia);

n_2 – parodo kurioje aptarnavimo bloko fazėje aptarnaujama paraiška (0 – jei aptarnavimo blokas tuščias).

1.2.2. G/M/1 SISTEMŲ MODELIAVIMAS

1.2.2.1. G/M/1 SISTEMOS MODELIS, KAI G – ERLANGO MIŠINYS

Nagrinėjama sistema *G/M/1*, kai bendrojo tipo skirstinio funkcija *G* aproksimuojama antrojo tipo antros eilės Erlango mišiniu . Aproksimuojamos sistemos fazinė schema:



1.4 pav. Sistemos G/M/1 aproksimavimo schema, kai G – Erlango mišinys

Sistema formaliai aprašoma įvykių kalba. Įvykių, kurie gali įvykti sistemoje, aibė yra $E=\{e_1,e_2,e_3,e_4,e_5,e_6,e_7,e_8\}$. Sistemos būsenų aibė: $N=\{(n_1,n_2,n_3)\}$, $n_1=\overline{0,5}$, $n_2=\overline{0,L}$, $n_3=0,1$ [10].

e_1 – paraiška atėjo į srauto bloką, pirmąją būseną;

e_2 – paraiška perėjo į srauto bloko antrą būseną;

e_3 – paraiška perėjo į srauto bloko trečią būseną;

e_4 – paraiška perėjo į srauto bloko ketvirtą būseną;

e_5 – paraiška perėjo į srauto bloko penktą būseną;

e_6 – paraiška atvyko į aptarnavimo įrenginį iš srauto bloko ketvirtos būsenos.

e_7 – paraiška atvyko į aptarnavimo įrenginį iš srauto bloko penktos būsenos.

e_8 – paraiška baigta aptarnauti aptarnavimo įrenginyje;

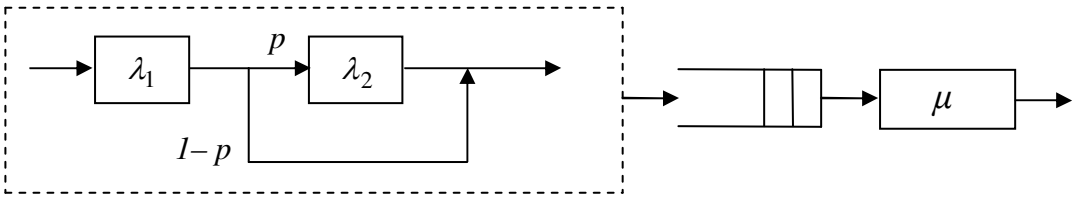
n_1 – parodo kurioje srauto bloko fazėje yra paraiška (0 – jei srauto blokas tuščias);

n_2 – paraiškų skaičius aptarnavimo sistemoje (eilėje ir aptarnavimo įrenginyje);

n_3 – paraiškų skaičius aptarnavimo įrenginyje (0 – jei aptarnavimo įrenginys tuščias, ir 1 – jei užimtas).

1.2.2.2. G/M/1 SISTEMOS MODELIS, KAI G – KOKSO SKIRSTINYS

Nagrinėjama sistema *G/M/1*, kai bendrojo tipo skirstinio funkcija *G* aproksimuojama Kokso skirstiniu. Aproksimuojamos sistemos fazinė schema:



1.5 pav. Sistemos G/M/1 aproksimavimo schema, kai G – Kokso skirstinys

Sistema formaliai aprašoma įvykių kalba. Įvykių, kurie gali įvykti sistemoje, aibė yra $E = \{e_1, e_2, e_3, e_4, e_5\}$. Sistemos būsenų aibė: $N = \{(n_1, n_2, n_3)\}$, $n_1 = \overline{0,2}$, $n_2 = \overline{0,L}$, $n_3 = 0,1$ [10].

e_1 – paraiška atėjo į srauto bloką, pirmąją fazę ;

e_2 – paraiška perėjo į srauto bloko antrą fazę su tikimybe p ir intensyvumu λ_1 ;

e_3 – paraiška baigta aptarnauti pirmoje srauto bloko fazėje ir ateina į aptarnavimo įrenginį su tikimybe $1 - p$ ir intensyvumu λ_1 ;

e_4 – paraiška baigta aptarnauti srauto bloko antroje fazėje ir ateina į aptarnavimo įrenginį su intensyvumu λ_2.

e_5 – paraiška baigta aptarnauti ir palieka sistemą su intensyvumu μ ;

n_1 – parodo kurioje srauto bloko fazėje yra atvykstanti paraiška (0 – jei srauto blokas tuščias);

n_2 – paraiškų skaičius sistemoje (0 – jei sistema tuščia) ;

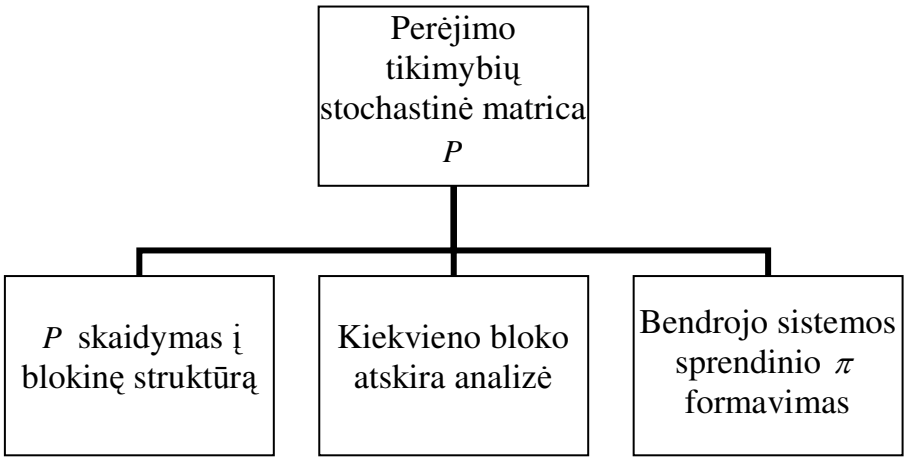
n_3 – paraiškų skaičius aptarnavimo įrenginyje (0 – jei aptarnavimo blokas tuščias).

1.3. DEKOMPOZICIJOS METODAS

Dekompozicijos metodas, kuris vadovaujasi principu „skaldyk ir valdyk“, gali būti panaudotas Markovo grandinių nagrinėjimui. Jeigu analizuojamas modelis yra didelės apimties, jis suskaidomas į subsistemas. Bendrasis sistemos sprendinys formuojamas iš dalinių sprendinių, kurie gaunami kiekvieną subsistemą nagrinėjant atskirai. Geriausia, kai problema suskaidoma į subproblemas, kurias galime išspręsti nepriklausomai viena nuo kitos, ir bendrasis sprendinys gaunamas tiesiog sujungiant visus atskirus sprendinius į vieną visumą.

Nors retai pasitaiko, kad Markovos grandinės gali būti suskaidomos į nepriklausomas subgrandines, tačiau įprasta, kad ši sąlyga beveik galioja. Markovo modeliuose dažnai pasitaiko atvejų, kai sistemos būsenų erdvė gali būti suskaidoma į nesusikertančius poaibius, kurie silpnai sąveikauja tarpusavyje, bet stiprūs sąveikos ryšiai galioja tarp būsenų poaibio viduje. Tokios problemos kartais vadinamos kaip beveik visiškai išskaidomos (*NCD* – nuo angliškų žodžių nearly completely decomposable). Akivaizdu, jog prielaida apie atskirų subsistemų nepriklausomą sprendimą negalioja. Todėl vienu ar kitu būdu paklaidos atsiranda. Todėl tikėsimės, kad šios paklaidos bus nedidelės, jei prielaida bus pasirinkta teisingai.

Dekompozicijos metodo taikymas Markovo grandinėse grafiškai pavaizduotas 1.6 paveiksle.



1.6 pav. Dekompozicijos metodo schema

Metodika grindžiama idėja [3,8], jog dideles sistemas lengva analizuoti, kai visos sistemos būsenos gali būti sugrupuotos į nedidelį skaičių grupių, kuriose:

1) Sąveikos tarp grupės būsenų gali būti nagrinėjamos taip, lyg sąveikos tarp grupių neegzistuotų;
2) Sąveikos tarp grupių gali būti nagrinėjamos nekreipiant dėmesio į sąveikas grupės viduje.

1.3.1. MATRICOS *P* BLOKINĖ STRUKTŪRA

Stipri sąveika tarp grupės būsenų ir silpna sąveika tarp pačių grupių [8] leidžia manyti, kad Markovo grandinės stochastinė perėjimo tikimybių matrica gali būti suvesta į blokinę formą

$$P = \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{pmatrix}, \qquad \textbf{(1.1)}$$

kurios nenuliniai elementai, nesantys įstrižainės blokuose, yra pakankamai maži lyginant su įstrižainės blokų elementais. Subblokai P_{ii} yra kvadratiniai, n_i-tosios eilės:

$$n = \sum_{i=1}^{N} n_i \ , \ i = 1,2,\ldots,N \qquad \textbf{(1.2)}$$

Sakysime, kad

$$\left\| P_{ii} \right\| = O(1) \ , \ i = 1,2,\ldots,N \ ,$$

$$\left\| P_{ij} \right\| = O(\varepsilon) \ , \ i \neq j \ ,$$

čia $\left\| \bullet \right\|$ žymi matricos spektrinę normą; ε - mažas teigiamas skaičius.

Visų pirma atliekama procedūra, kuri leidžia kuo tiksliau nustatyti, kaip Markovo grandinė turi būti suskirstyta į blokinę struktūrą. Ją sudaro trys pagrindiniai etapai (1.1 lentelė).

1.1 lentelė

Blokinės struktūros paieška

Etapas	Veiksmas
1.	Įvedamas dekompozicijos parametras γ, kuris nuo 10^{-10} iki 10^{-1}
2.	Visi matricos P elementai, mažesni ar lygūs parametrui γ, pakeičiami nuliais
3.	Naujoji matrica traktuojama kaip orientuoto grafo vaizdavimas ir vykdoma stipriai sujungtų grafo viršūnių paieška [1]

Atlikus veiksmų seką, rezultatas yra matrica P, tačiau jau turinti blokinę struktūrą (1.1). Kaip jau buvo minėta anksčiau, sistemos būsenų stacionarių tikimybių vektorius apibrėžiamas kaip sistemos

$$\pi \cdot P = \pi \qquad \textbf{(1.3)}$$

sprendinys. Toliau nagrinėsime atskirų sistemų P_{ii} sprendinius.

1.3.2. BLOKŲ P_{ii} SPRENDINIAI

Tegu stacionarių tikimybių vektorius π skaidomas į n_i ilgio vektorius π_i atitinkamai kaip matrica P:

$$\pi = (\pi_1, \pi_2, \ldots, \pi_N). \tag{1.4}$$

Jeigu visi blokai už įstrižainės ribų lygūs nuliui, tuomet matrica P yra visiškai išskaidoma. Iš (1.1), (1.3) ir (1.4) gauname, kad

$$(\pi_1, \pi_2, \ldots, \pi_N)\begin{pmatrix} P_{11} & 0 & \cdots & 0 & 0 \\ 0 & P_{22} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & P_{N-1N-1} & 0 \\ 0 & 0 & \cdots & 0 & P_{NN} \end{pmatrix} = (\pi_1, \pi_2, \ldots, \pi_N). \tag{1.5}$$

Tuomet kiekvienas vektorius π_i gali būti apskaičiuotas iš

$$\pi_i \cdot P_{ii} = \pi_i \tag{1.6}$$

Bendru atveju, kai blokai neįstrižainėje turi nenulinių elementų, darysime prielaidą, jog sistema yra visiškai išskaidoma, ir kiekvienam blokui stacionarias tikimybes skaičiuosime pagal (1.6). Dėl šios prielaidos iškyla pirmoji problema: P_{ii} nėra stochastinė, bet griežtai substochastinė matrica (eilučių suma nėra lygi vienetui). Tačiau literatūroje [9] rekomenduojama dirbti su substochastinėmis matricomis. Kitaip tariant, galime naudoti normuotą tikrinį vektorių u_i ($\|u_i\|_1 = 1$), atitinkantį bloko P_{ii} Perron šaknį (matricos P_{ii} tikrinė reikšmė arčiausiai vieneto), kaip bloko i sąlygines tikimybes. Tokiu būdu vektoriaus u_i k-tasis elementas aproksimuoja buvimo bloko i k-tojoje būsenoje tikimybę, esant sąlygai, kad sistema pereina į vieną iš bloko i būsenų. Todėl kiekvienam blokui i, $1 \le i \le N$, galioja

$$u_i P_{ii} = \lambda_i u_i,\ u_i e = 1, \tag{1.7}$$

čia λ_i - bloko P_{ii} Perron šaknis, u_i - tikrinis vektorius, kurio ilgis n_i, $e = (1,1,\ldots,1_i)^T$.

Tokiu būdu apskaičiuojame (aproksimuotus) stacionarių tikimybių vektorius kiekvienam blokui.

1.3.3. BENDRASIS SISTEMOS SPRENDINYS π

Apskaičiavę kiekvieno bloko stacionarias tikimybes, susiduriame su kita problema – paprastai sujungus visus gautus vektorius į vieną, negausime tikimybės vektoriaus. Kodėl? Kiekvieno subvektoriaus elementų suma lygi vienetui, todėl akivaizdu, kad naujam vektoriui ši sąlyga nebegalioja. Todėl kiekvienam subvektoriui įvedamas svoris – tikimybė būti viename iš blokų:

$$\left(\|\pi_1\|_1, \|\pi_2\|_1, \ldots, \|\pi_N\|_1\right). \qquad \textbf{(1.8)}$$

Dydis $\|\pi_i\|_1$ parodo tikimybę, kad sistema yra vienoje iš bloko i būsenų. Anksčiau minėjome, kad u_i yra sąlyginės tikimybės būti konkrečioje bloko būsenoje su sąlyga, kad sistema yra vienoje iš bloko būsenų. Daugindami kiekvieną vektorių u_i iš $\|\pi_i\|_1$ šią sąlygą panaikiname.

Remiantis Stewart (1994), formuojame $N \times N$ dydžio matricą A (žymėjimas nuo angliško žodžio *aggregation*):

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{pmatrix}. \qquad \textbf{(1.9)}$$

Kiekvienas matricos A elementas

$$a_{ij} = \frac{\pi_i}{\|\pi_i\|} P_{ij} e = \varphi_i P_{ij} e \qquad \textbf{(1.10)}$$

parodo tikimybę, kad sistema, palikdama vieną iš bloko i būsenų, pereis į vieną iš bloko j būsenų (analogiškai matricos P atveju). Formulę (1.10) panagrinėkime išsamiau. Matricos $P_{ij}e$ (dydis $n \times N$) elementai yra ne kas kita kaip blokų P_{ij} kiekvienos eilutės elementų suma, t.y. k-tosios bloko P_{ij} eilutės suma parodo tikimybę, kad sistema, palikdama bloko i k-tąją būseną, pereis į vieną iš bloko j būsenų. Daugindami $P_{ij}e$ iš $\varphi_i = \pi_i / \|\pi_i\|_1$ panaikiname sąlygą, jog sistema palieka būtent k-tąją būseną. Todėl matrica A charakterizuoja sąveikas tarp blokų.

Kadangi A yra baigtinė stochastinė matrica (teorema literatūroje [9]), todėl galime kalbėti apie naują stacionarių tikimybių vektorių, kurį žymėsime $\xi = \left(\xi_1, \xi_2, \ldots, \xi_N\right)$. Analogiškai (1.3) atveju, ξ apibrėžiamas kaip sistemos (1.11) sprendinys:

$$\xi \cdot A = \xi\,,\ \xi e = 1\,. \qquad \textbf{(1.11)}$$

Taigi, i-tasis vektoriaus ξ elementas yra stacionari tikimybė, kad sistema yra bloke i (bet kurioje iš būsenų). Be to, į (1.11) formulę vietoj ξ įstačius (1.8) vektorių bei atlikus elementarius veiksmus [9], gauname teisingą lygybę. Todėl akivaizdu, kad

$$\xi = \left(\|\pi_1\|_1, \|\pi_2\|_1, \ldots, \|\pi_N\|_1\right). \qquad \textbf{(1.12)}$$

Turėdami kiekvieno bloko stacionarias tikimybes π_i iš (1.6) bei svorius ξ_i iš (1.11), galime suformuoti visos sistemos sprendinį π:

$$\pi = (\xi_1\pi_1, \xi_2\pi_2, \ldots, \xi_N\pi_N)\,. \qquad \textbf{(1.13)}$$

Kita vertus, π yra ieškomas dydis, todėl neįmanoma surasti dydžių $\|\pi_i\|_1$ tiesiogini būdu.

1.3.4. NCD APROKSIMACIJA – RAYLEIGH-RITZ TIKSLINIMO ŽINGSNIS

Iš ankstesnio skyrelio matome, jog jeigu suformuojame matricą A, tai galime apibrėžti jos stacionarių tikimybių vektorių. Tokiu būdu surandame subvektorių u_i svorius. Atrodytų, kad jau turime pilną sprendimo ciklą, bet matricos A formavimui reikia žinoti $\varphi_i = \pi_i / \|\pi_i\|_1$. Tačiau turėdami subvektorius u_i kaip vektorių π_i aproksimacijas, galime juos panaudoti pastarosios problemos sprendimui. Tarsime, kad

$$\varphi_i = \frac{\pi_i}{\|\pi_i\|_1} \approx \frac{u_i}{\|u_i\|_1}, \qquad \textbf{(1.14)}$$

Todėl dabar matrica A apskaičiuojama pagal

$$\left(A^*\right)_{ij} = \frac{u_i}{\|u_i\|} P_{ij} e \qquad \textbf{(1.15)}$$

Atitinkamai galime įvertinti svorius $\xi = (\xi_1, \xi_2, \ldots, \xi_N)$ bei apytiksliai apskaičiuoti stacionarių tikimybių vektorių $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$. Pilna sprendimo schema, įgyvendinanti minėtą veiksmų seką, aprašyta 1.2 lentelėje.

1.2 lentelė

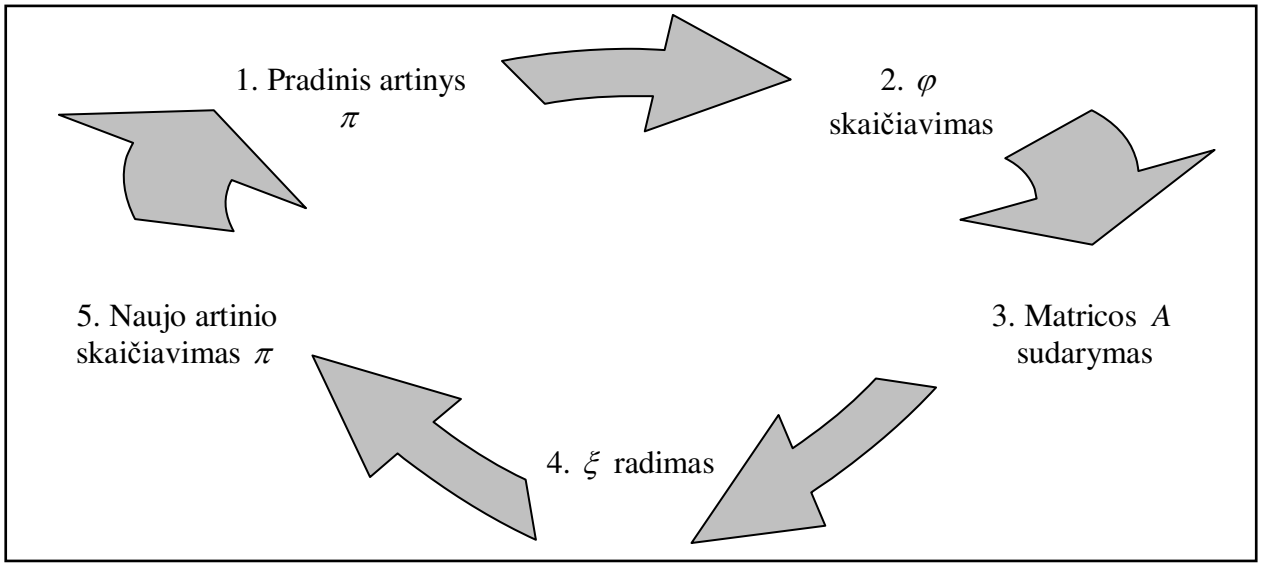
Rayleigh-Ritz tikslinimo žingsnis

Etapas	Veiksmas
1.	Kiekvieno matricos P bloko analizė. Skaičiuojamas vektorius u_i: $u_i P_{ii} = \lambda_i u_i,\ u_i e = 1.$
2.	Sudaroma matrica A^*: $A^* = \begin{pmatrix} u_1 & 0 & \cdots & 0 \\ 0 & u_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_N \end{pmatrix} \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{pmatrix} \begin{pmatrix} e & 0 & \cdots & 0 \\ 0 & e & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e \end{pmatrix}.$
3.	Skaičiuojamas vektorius ξ^*: $\xi^* A^* = \xi^*,\ \xi^* e = 1.$
4.	Sudaromas apytikslis stacionarių tikimybių vektorius π^*: $\pi^* = (\xi_1^* u_1, \xi_2^* u_2, \ldots, \xi_N^* u_N).$

Dėl panašumo į Rayleigh-Ritz procedūrą, kuri skaičiuoja apytiksles matricos tikrines reikšmes bei tikrinius vektorius [9], 1.2 lentelėje pateikta NCD aproksimacija dažnai vadinama Rayleigh-Ritz tikslinimo žingsniu.

1.3.5. ITERACINIAI METODAI

Atlikus veiksmų seką, aprašytą 1.3.2 – 1.3.4 poskyriuose, rezultatas yra apytikslis stacionarių tikimybių vektorius π^*. Automatiškai kyla klausimas, ar dar kartą panaudojus gautą sprendinį antrame Rayleigh-Ritz žingsnyje, negausime tikslesnio rezultato. Pasirodo, kad toks sprendimo variantas visiškai įmanomas, nes vektoriaus u_i pakeitimas vektoriumi $\xi_i^* u_i$ neturi jokios neigiamos įtakos stacionarių tikimybių skaičiavimui [9]. Kitaip sakant, Rayleigh-Ritz žingsnis yra stacionarus.



1.7 pav. IAD algoritmų idėja

Tokią sprendimo idėją įgyvendina *IAD* (nuo angliškų žodžių *iterative aggregation/disaggregation*) algoritmai.

1.3.5.1. KMS ALGORITMAS

Vienas iš *IAD* šeimos algoritmų yra *KMS* (*Koury, McAllister ir Stewart*) algoritmas, kurio pagrindiniai žingsniai pavaizduoti 1.1 lentelėje.

1.3 lentelė

KMS algoritmas

Žingsnis	Veiksmas
1.	Pradinis artinys: $\pi^{(0)} = \left(\pi_1^{(0)}, \pi_2^{(0)}, \ldots, \pi_N^{(0)}\right),\ m = 1.$
2.	$\varphi_i^{(m-1)} = \dfrac{\pi_i^{(m-1)}}{\left\|\pi_i^{(m-1)}\right\|_1},\ i = 1,2,\ldots N.$
3.	$\left(A^{(m-1)}\right)_{ij} = \varphi_i^{(m-1)} P_{ij} e.$
4.	$\xi^{(m-1)} A^{(m-1)} = \xi^{(m-1)},\ \xi^{(m-1)} e = 1.$
5. (a)	$z^m = \left(\xi_1^{(m-1)}\varphi_1^{(m-1)}, \xi_2^{(m-1)}\varphi_2^{(m-1)}, \ldots, \xi_N^{(m-1)}\varphi_N^{(m-1)}\right)$
5. (b)	$\pi_k^{(m)} = \pi_k^{(m)} P_{kk} + \sum_{j<k} \pi_j^{(m)} P_{jk} + \sum_{j>k} z_j^{(m)} P_{jk},\ k = 1,2,\ldots N.$
6.	Jei tikslumas tenkina, tai sprendinys $\pi^{(m)}$. Kitu atveju $m = m+1.$

Iš lentelės matyti, jog žingsniai 1 – 5a yra lygiai tokie patys kaip ir Rayleigh-Ritz tikslinimo žingsnyje. Tačiau jeigu 5a žingsnyje gautas rezultatas bus grąžinamas į 2-ą poziciją, tai nauja matrica A (kartu ir apytikslis sprendinys) bus lygiai tokia pat. Todėl 5b veiksmas patikslina gautą sprendinį žingsnyje 5a bei leidžia suformuoti naują matricą A. Toks sprendinio tikslinimas pagreitina algoritmo konvergavimą prie galutinio rezultato. 5b žingsnyje reikia spręsti N lygčių sistemą, kurią galime užrašyti įprasta forma:

$$Bx = r,$$

čia $B = \left(I - P_{kk}\right)^T,\ x^T = \pi_k^{(m)},\ r^T = \sum_{j<k} \pi_j^{(m)} P_{jk} + \sum_{j>k} z_j^{(m)} P_{jk},\ k = 1,2,\ldots N.$

1.3.5.2. TAKAHASHI ALGORITMAS

Antrasis iš IAD algoritmų grupės yra *Takahashi* algoritmas, kuris dažnai apibūdinamas kaip sąlyginių tikimybių kintamųjų metodas [9]. Kaip jau buvo minėta anksčiau, jeigu turime vektorių $\varphi = \left(\varphi_1, \varphi_2, \ldots, \varphi_N\right)$, tai suformavę matricą A ir radę vektorių ξ iš (1.11), apskaičiuojame sprendinį $\pi = \left(\xi_1\varphi_1, \xi_2\varphi_2, \ldots, \xi_N\varphi_N\right)$. *Takahashi* metodo idėja yra ta, kad kiekvieną subvektorių φ_k galime apskaičiuoti turėdami vektorių ξ ir subvektorius $\varphi_i,\ i \neq k$. Iteracinis algoritmas iš apytikslaus vektoriaus $\varphi^{(m-1)}$ visų pirma apskaičiuoja $\xi^{(m-1)}$ bei naują vektorių $\varphi^{(m)}$. Subvektorius $\varphi_k^{(m)}$ apskaičiuojamas iš $\xi^{(m)}$, $\varphi_i^{(m)}$ (kai $i < k$) ir $\varphi_i^{(m-1)}$ (kai $i > k$) (plačiau literatūroje [9]). Algoritmo struktūra pateikta 1.2 lentelėje.

1.4 lentelė

Takahashi algoritmas

Žingsnis	Veiksmas
1.	Pradinis artinys: $\pi^{(0)} = \left(\pi_1^{(0)}, \pi_2^{(0)}, \ldots, \pi_N^{(0)}\right),\ m = 1.$
2.	$\varphi_i^{(m-1)} = \dfrac{\pi_i^{(m-1)}}{\left\|\pi_i^{(m-1)}\right\|_1},\ i = 1,2,\ldots N.$
3.	$\left(A^{(m-1)}\right)_{ij} = \varphi_i^{(m-1)} P_{ij} e.$
4.	$\xi^{(m-1)} A^{(m-1)} = \xi^{(m-1)},\ \xi^{(m-1)} e = 1.$
5.	$z_k^{(m)} = z_k^{(m)} P_{kk} + \sum_{j<k} \xi_j^{(m-1)} \varphi_j^{(m)} P_{jk} + \sum_{j>k} \xi_j^{(m-1)} \varphi_j^{(m-1)} P_{jk},$ $\varphi_k^{(m)} = \dfrac{z_k^{(m)}}{\left\|z_k^{(m)}\right\|_1},\ k = 1,2,\ldots N$
6.	$\pi^{(m)} = \left(\xi_1^{(m-1)} \varphi_1^{(m)}, \xi_2^{(m-1)} \varphi_2^{(m)}, \ldots, \xi_N^{(m-1)} \varphi_N^{(m)}\right),\ k = 1,2,\ldots N.$
7.	Jei tikslumas tenkina, tai sprendinys $\pi^{(m)}$. Kitu atveju $m = m + 1.$

Matome, kad *Takahashi* algoritmas savo struktūra labai panašus į *KMS* algoritmą. Esminis skirtumas egzistuoja tik penktame žingsnyje.

2. TIRIAMOJI DALIS

Realizavus metodiką, kurią aptarėme teorinėje dalyje, ištirsime dekompozicijos metodo galimybes aptarnavimo sistemų modeliuose *M/G/1* ir *G/M/1*. Kadangi perėjimo tikimybių matricos P elementų išsidėstymas nepriklauso nuo bendrojo skirstinio *G* tipo, todėl tiriamosios dalies įgyvendinimui pasirenkame, kad *G* - Veibulo skirstinys, kurio pasiskirstymo funkcija:

$$p(t) = 1 - \exp\left\{-\frac{t}{\lambda}\right\}^k,\ t \ge 0,\ k > 0,\ \lambda > 0$$

Pasirenkame parametrus $k = 0{,}7$, $\lambda = 0{,}7$. Tuomet Veibulo skirstinio 3 pradiniai momentai:

$$m_1 = 0.8861,\ m_2 = 2.4643,\ m_3 = 12.7713.$$

Antrojo tipo Erlango mišinio parametrai:

$$n = 1; \quad p = 0.3106; \quad \mu_1 = 0.5313; \quad \mu_2 = 2.2869$$

Kokso skirstinio parametrų reikšmės, sulyginant 3 pradinius momentus:

$$p = 0.2385; \quad \mu_1 = 2.2869; \quad \mu_2 = 0.5313.$$

Visų pirma panagrinėsime aptarnavimo sistemas, kuriose paraiškų skaičius nėra didelis. Tokiais atvejais gaunamos mažo matavimo perėjimo tikimybių matricos P.

2.1. APTARNAVIMO SISTEMŲ M/G/1 TYRIMAS

2.1.1. M/G/1 SISTEMOS TYRIMAS, KAI G – ERLANGO MIŠINYS

Nagrinėkime aptarnavimo sistemos modelį *M/G/1*, kai skirstinio funkcija *G* aproksimuojama Erlango mišiniu (1.2.1.1 skyrelis). 2.1 lentelėje pateiktas sistemos aprašymo įvykių kalba pseudokodas. Sumodeliavus sistemą, kurioje maksimalus paraiškų skaičius $L = 4$, gauname $n = 13$ eilės perėjimo tikimybių matricą:

$$P = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1e-05 & 0.31 & 0.69 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1e-05 & 0.31 & 0.69 & 0 & 0 & 0 & 0 & 0 \\
0.35 & 0 & 0 & 0 & 0 & 0 & 0.65 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1e-05 & 0.31 & 0.69 & 0 & 0 \\
0 & 0.35 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.65 & 0 & 0 & 0 \\
0 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.31 & 0.69 \\
0 & 0 & 0.35 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.65 & 0 \\
0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \quad \textbf{(2.1)}$$

2.1 lentelė

M/G/1 sistemos pseudokodas (G – Erlango mišinys)

e1: if $n_2 = 0$ and $n_1 < L$ then $n_1 \leftarrow n_1 + 1;\ n_2 \leftarrow 1;$ else if $n_1 < L$ then $n_2 \leftarrow n_2 + 1;$ end if end if Return Intens $\leftarrow \lambda$	e2: if $n_2 = 1$ then $n_2 \leftarrow 2$ end if Return Intens $\leftarrow p \cdot C$
e3: if $n_2 = 1$ then $n_2 \leftarrow 3$ end if Return Intens $\leftarrow (1-p) \cdot C$	e4: if $n_2 = 2$ then $n_2 \leftarrow 4$ end if Return Intens $\leftarrow \mu_1$
e5: if $n_1 = 3$ then $n_1 \leftarrow 5$ end if Return Intens $\leftarrow \mu_2$	e6: if $n_1 = 4$ if $n_2 > 1$ then $n_1 \leftarrow n_1 - 1;\ n_2 \leftarrow 1;$ else $n_2 \leftarrow n_2 + 1;\ n_2 \leftarrow 0$ end if end if Return Intens $\leftarrow \mu_1$
e7: if $n_1 = 5$ if $n_2 > 1$ then $n_1 \leftarrow n_1 - 1;\ n_2 \leftarrow 1;$ else $n_2 \leftarrow n_2 + 1;\ n_2 \leftarrow 0$ end if end if Return Intens $\leftarrow \mu_2$	

Įvykiuose *e2* ir *e3* naudojamas simbolis *C* – pakankamai didelis teigiamas skaičius. Skaičius *C* įvedamas tam, kad paraiškų buvimo laikas fiktyvioje fazėje būtų daug kartų mažesnis negu fazėse su eksponentiniais skirstiniais. Praktinis sistemų modeliavimas parodė, kad toks modeliavimas (t.y., įvedant fiktyvią fazę) duoda teisingus rezultatus. Konstantos *C* parinkimas priklauso nuo kitų sistemos būsenų intensyvumų. Praktikoje galima naudoti tokį parinkimo metodą – konstanta *C* 100000 kartų didesnė už didžiausią iš likusių sistemos intensyvumų.

Kompiuteriniai skaičiavimai atliekami 15 skaitmenų po kablelio tikslumu, tačiau dėl patogumo pateiksime suapvalintus rezultatus (atsižvelgiant į pavaizdavimo galimybes apvalinimas gali skirtis).

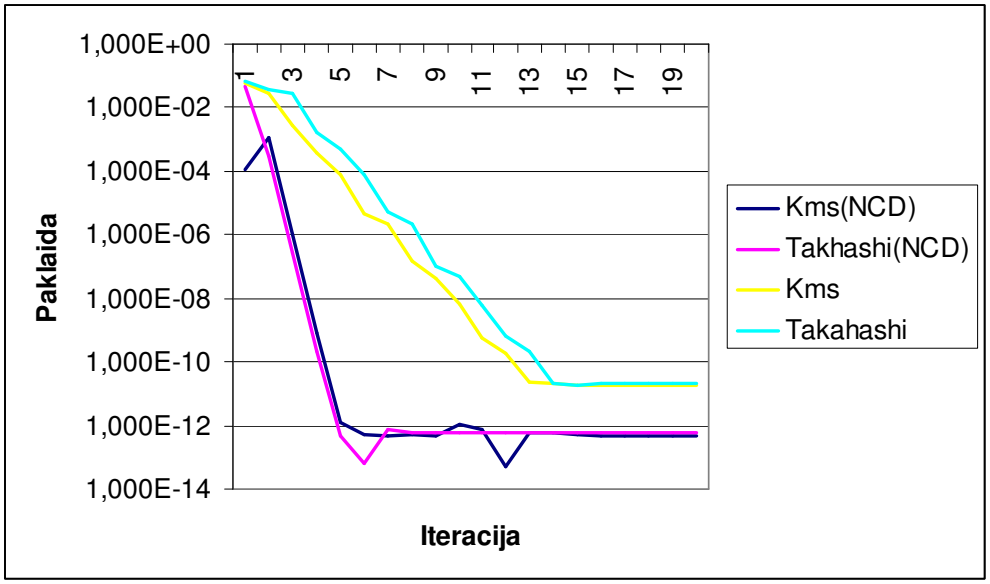
Kaip matome iš (2.1), matricos elementų išsidėstymas turi tam tikrą struktūrą: elementai koncentruojasi ne pagrindinėje įstrižainėje (vėliau buvo pastebėta, kad toks išsidėstymas būdingas modeliui *M/G/1*, kai *G* – Erlango mišinys). Be to, beveik visos perėjimo tikimybės didesnės už 0,1, todėl blokinės struktūros paieška (1.1 lentelė) šiuo atveju netinka, nes programa neranda mažesnių matricų, kurios tenkintų reikiamas sąlygas. Pabandykime pažiūrėti, kaip *IAD* algoritmai veiks skirtingai padalijus matricą P bei palyginkime su įprastu lygčių sistemos sprendimo variantu.

2.2 lentelė

M/G/1 modelio paklaidų palyginimas (G – Erlango mišinys)

Blokų dydžiai	Paklaida	
	KMS	Takahashi
(5, 3, 5)	1,0415E-11	5,0739E-10
(4, 4, 5)	9,0129E-09	1,6161E-07
(2, 5, 3, 3)	5,2137E-07	2,7711E-06
(6, 3, 2, 2)	2,5638E-10	3,5354E-9
(7,6)	1,5845E-11	1,4806E-11

Lentelėje 2.2 pateikti skaičiavimo rezultatai gauti atliekant po 20 iteracijų (*KMS* ir *Takahashi* algoritmuose). Kadangi matricos P padalijimas į submatricas, kurių elementai netenkina teorinės dalies reikalavimų, konvergavimas nėra toks greitas. Palyginimui pateikiame *NCD* sistemos ir nagrinėto atvejo (matrica skaidoma į P_{77} ir P_{66}) konvergavimo grafikus.



2.1 pav. NCD ir modelio M/G/1 (G – Erlango mišinys) konvergavimo palyginimas

Iš 2.1 paveikslo matyti, kaip skiriasi konvergavimo greitis, kai nutolstame nuo metodikos reikalavimų. Idealiu atveju stacionarių tikimybių skaičiavimui atlikti iki 6 iteracijų, kai tuo tarpu dabar užtrunkame dvigubai ilgiau. Iš kitos pusės, nereikalaudami didelio tikslumo galime sutaupyti dalį laiko.

2.1.2. M/G/1 SISTEMOS TYRIMAS, KAI G – KOKSO SKIRSTINYS

Nagrinėkime aptarnavimo sistemos modelį *M/G/1*, kai skirstinio funkcija *G* aproksimuojama Kokso skirstiniu(1.2.1.2 skyrelis). 2.3 lentelėje pateiktas sistemos aprašymo įvykių kalba pseudokodas.

2.3 lentelė

M/G/1 sistemos pseudokodas (G – Kokso skirstinys)

e1: if $n_2 = 0$ and $n_1 < L$ then $n_1 \leftarrow n_1 + 1;\ n_2 \leftarrow 1;$ else if $n_1 < L$ then $n_2 \leftarrow n_2 + 1;$ end if end if Return Intens $\leftarrow \lambda$	e2: if $n_2 = 1$ then $n_2 \leftarrow 2$ end if Return Intens $\leftarrow p \cdot \mu_1$
e3: if $n_2 = 1$ if $n_1 > 1$ then $n_1 \leftarrow n_1 - 1;\ n_2 \leftarrow 1;$ else $n_1 \leftarrow 0;\ n_2 \leftarrow 0$ end if end if Return Intens $\leftarrow (1-p) \cdot \mu_1$	e4: if $n_2 = 2$ if $n_2 > 1$ then $n_1 \leftarrow n_1 - 1;\ n_2 \leftarrow 1;$ else $n_1 \leftarrow 0;\ n_2 \leftarrow 0$ end if end if Return Intens $\leftarrow \mu_2$

Sumodeliavus sistemą, kurioje maksimalus paraiškų skaičius $L = 5$, gauname $n = 11$ eilės perėjimo tikimybių matricą:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.53 & 0 & 0.3 & 0.17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.53 & 0 & 0 & 0.3 & 0.17 & 0 & 0 & 0 & 0 & 0 \\ 0.35 & 0 & 0 & 0 & 0 & 0.65 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.53 & 0 & 0 & 0 & 0.3 & 0.17 & 0 & 0 & 0 \\ 0 & 0.35 & 0 & 0 & 0 & 0 & 0 & 0.65 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.53 & 0 & 0 & 0 & 0.3 & 0.17 & 0 \\ 0 & 0 & 0.35 & 0 & 0 & 0 & 0 & 0 & 0 & 0.65 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.76 & 0 & 0 & 0 & 0.24 \\ 0 & 0 & 0 & 0 & 0.35 & 0 & 0 & 0 & 0 & 0 & 0.65 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \textbf{(2.2)}$$

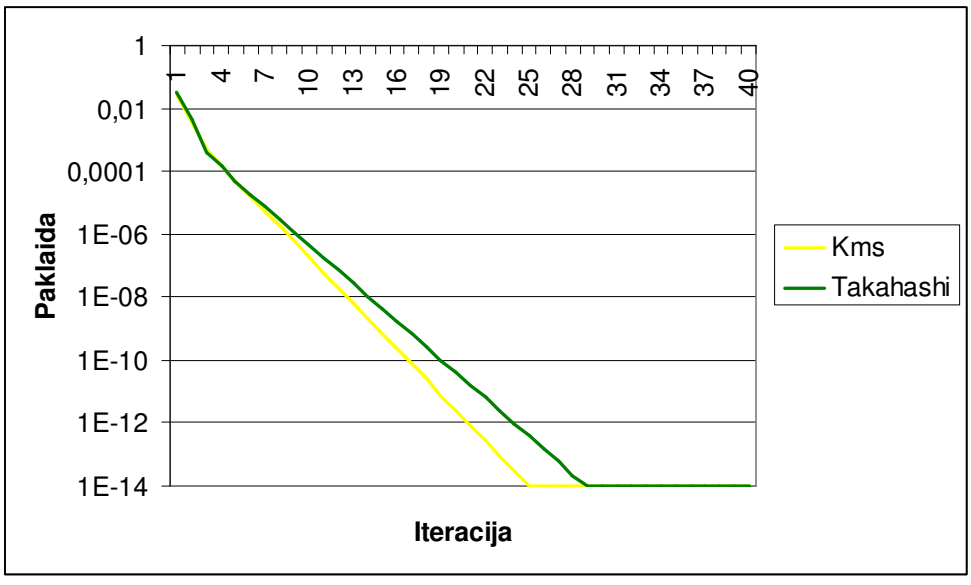
Vėlgi akivaizdžiai matome, jog matricos P (2.2) elementai didesni už 0,1. Todėl dekompozicijos parametro keitimas ir šiuo atveju jokios naudos neduoda. Tačiau šįkart galime išskirti bent vieną submatricą, kuri tenkina užsibrėžtą sąlygą (stipriai sujungtos grafo viršūnės). Tai galėtų būti blokai P_{22}, P_{33} ir kiti. Parinkus pirmą bloką tokiu būdu, tolimesnis procesas sustoja, nerasdamas naujų skaidymo galimybių. Kadangi tyrimo eigoje pastebėta, kad matricos P skaidymas į didesnės eilės submatricas duoda geresnį rezultatą, tai šiuo atveju dekompozicija (3, 4, 4) pasirodė efektyviausia.

2.4 lentelė

M/G/1 sistemos sprendinys (G – Kokso skirstinys)

Įprastas būdas	IAD algoritmas		Paklaida	
	Kms	Takahashi	Kms	Takahashi
0,13882885603994	0,13882885603681	0,13882885610504	3,1300E-12	6,5100E-11
0,23634512607575	0,23634512607246	0,23634512616536	3,2900E-12	8,9610E-11
0,15088999540252	0,15088999540430	0,15088999541749	1,7800E-12	1,4970E-11
0,03920729296471	0,03920729296416	0,03920729297287	5,5000E-13	8,1600E-12
0,11492083384147	0,11492083384563	0,11492083379121	4,1600E-12	5,0260E-11
0,05063467986207	0,05063467986200	0,05063467986560	7,0000E-14	3,5300E-12
0,09737469374528	0,09737469375123	0,09737469367048	5,9500E-12	7,4800E-11
0,05213018211629	0,05213018211694	0,05213018211176	6,5000E-13	4,5300E-12
0,02962819806588	0,02962819806769	0,02962819804397	1,8100E-12	2,1910E-11
0,05019606077280	0,05019606077421	0,05019606075790	1,4100E-12	1,4900E-11
0,03984408111329	0,03984408111464	0,03984408109834	1,3500E-12	1,4950E-11

Rezultatai panašūs kaip ir Erlango mišinio atveju, t.y. tikslumas stabilizuojasi vienuolika-dvylika skaitmenų po kablelio. Iteracijų skaičius taipogi 20. Padidinus šį skaičių, galime tikėtis tikslesnio sprendinio. Tai patvirtina tyrimo rezultatai, kai iteracijų skaičius programoje padvigubinamas, t.y. iki 40-ies. Panagrinėkime šią situaciją išsamiau.



2.2 pav. IAD algoritmų palyginimas konvergavimo greičio prasme M/G/1 modelyje

Kaip matome iš 2.2 paveikslo, iteracijų skaičiaus padvigubinimas pasiteisino tik dalinai. Taip, pasiekėme didesnio tikslumo, tačiau pažiūrėkime, ką praradome. Nepriartėjus net iki 30-os iteracijos, metodų konvergavimas stabilizavosi ties *1E-14*. Šiuo atveju *KMS* algoritmas nežymiai lenkia *Takahashi*, tačiau tikrai ne tiek, kad būtų galima kalbėti apie jo neginčijamą pranašumą. Tiesa, pastaroji tendencija išryškėjo ir ankstesniuose pavyzdžiuose.

2.2. APTARNAVIMO SISTEMŲ G/M/1 TYRIMAS

2.2.1. G/M/1 SISTEMOS TYRIMAS, KAI G – ERLANGO MIŠINYS

Nagrinėkime aptarnavimo sistemos modelį *G/M/1*, kai skirstinio funkcija *G* aproksimuojama Erlango mišiniu (1.2.2.1 skyrelis). 2.6 lentelėje pateiktas sistemos aprašymo įvykių kalba pseudokodas. Sumodeliavus sistemą, kurioje maksimalus paraiškų skaičius $L = 2$, gauname $n = 12$ eilės perėjimo tikimybių matricą (tikimybės suapvalintos):

$$P = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.31 & 0.69 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1e-05 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1e-05 & 0 & 0 & 0 & 0 & 0.31 & 0.69 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.65 & 0 & 0 & 0 & 0 & 0 & 0.35 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.3 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1e-05 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1e-05 & 0 & 0 & 0 & 0 & 0.31 & 0.69 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{pmatrix} \qquad \textbf{(2.3)}$$

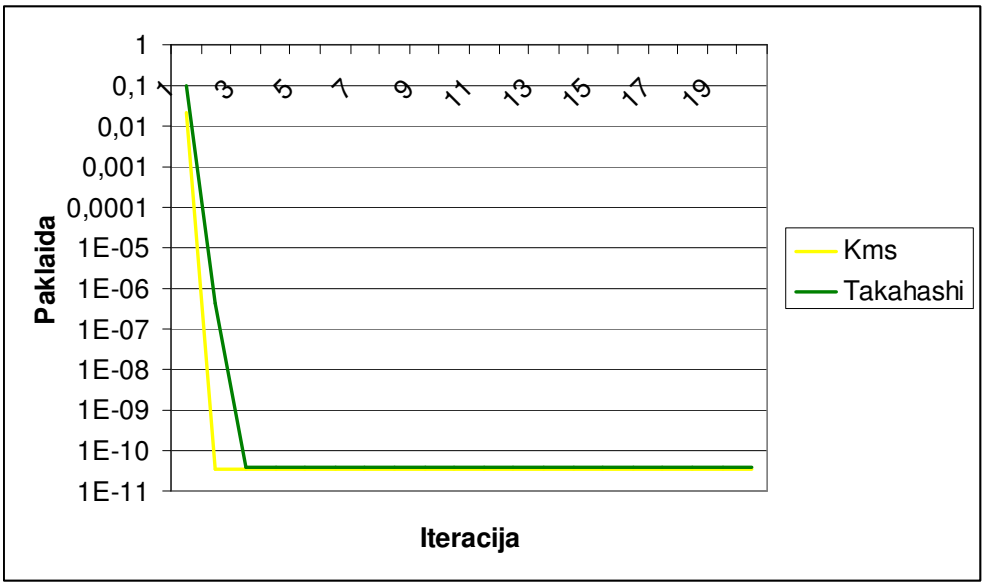
Kaip ir ankstesniais atvejais, matricos P (2.3) elementai išsidėstę virš ir žemiau pagrindinės įstrižainės. Taip pat dekompozicijos parametro keitimas šioje situacijoje nepadeda, nes eilinį kartą beveik visos tikimybės didesnės už 0,1. Tačiau analizuojant pradinę matricą, galime išskirti 2 blokus P_{11} ir P_{22}, kurie pažymėti raudona punktyrine linija. Tokiu būdu analizė persikelia į submatricas atskirai. Pažiūrėkime, ar rezultatai gaunami geresni, kai gana gerai išpildomos reikiamos sąlygos. 2.5 lentelėje matome, kad neišryškėjo labai pastebimas pranašumas paklaidų atžvilgiu, kad ir tiksliai suradome matricos skaidymą į submatricas. Tiek anksčiau, tiek dabar atlikus 20 iteracijų *IAD* algoritmuose sprendinio tikslumas svyruoja panašiose ribose.

2.5 lentelė

G/M/1 sistemos sprendinys (G – Erlango skirstinys)

Įprastas būdas	IAD algoritmas		Paklaida	
	Kms	Takahashi	Kms	Takahashi
0,00000103151505	0,00000103151506	0,00000103151506	1E-14	1E-14
0,00000206303448	0,00000206303448	0,00000206303448	0	0
0,05071014149816	0,05071014151084	0,05071014151084	1,268E-11	1,268E-11
0,05244092704807	0,05244092704597	0,05244092704597	2,1E-12	2,1E-12
0,10315253702087	0,10315253703144	0,10315253703144	1,057E-11	1,057E-11
0,10315297395544	0,10315297396602	0,10315297396602	1,058E-11	1,058E-11
0,07765263571203	0,07765263569380	0,07765263573144	1,823E-11	1,941E-11
0,17234530125459	0,17234530116413	0,17234530124767	9,046E-11	6,9199E-12
0,14684893145486	0,14684893138560	0,14684893145678	6,926E-11	1,9199E-12
0,14684746296554	0,14684746289629	0,14684746296747	6,925E-11	1,9301E-12
0,04561229047173	0,04561229045022	0,04561229047232	2,151E-11	5,9E-13
0,10123370406918	0,10123370397144	0,10123370402051	9,774E-11	4,867E-11

2.3 paveiksle pavaizduotas analizuojamo modelio, kai jis suskaidomas į 2 submodelius, algoritmo konvergavimas. Iš karto pastebime, jog konvergavimo greitis itin didelis – jau trečioje iteracijoje staigiai pasiekiamas maksimalus tikslumas (išsamesni tyrimai parodė, kad šiuo atveju didesnis iteracijų skaičius neturi jokios įtakos tikslumui). Palyginus 2.3 paveikslą su 2.1 paveikslo grafikais, kurie parodo *NCD* sistemos konvergavimą, mažas iteracijų skaičius būdingas abejais atvejais. *KMS* algoritmas vėlgi šiek tiek pranašesnis už *Takahashi*.



2.3 pav. IAD algoritmų palyginimas konvergavimo greičio prasme G/M/1 modelyje

2.6 lentelė

G/M/1 sistemos pseudokodas (G – Erlango mišinys)

e1: if $n_1 = 0$ then $n_1 \leftarrow 1$ end if Return Intens $\leftarrow C$	e2: if $n_1 = 1$ then $n_1 \leftarrow 2$ end if Return Intens $\leftarrow p \cdot C$
e3: if $n_1 = 1$ then $n_1 \leftarrow 3$ end if Return Intens $\leftarrow (1-p) \cdot C$	e4: if $n_1 = 2$ then $n_1 \leftarrow 4$ end if Return Intens $\leftarrow \lambda_1$
e5: if $n_1 = 3$ then $n_1 \leftarrow 5$ end if Return Intens $\leftarrow \lambda_2$	e6: if $n_1 = 4$ if $n_3 = 0$ and $n_2 < L$ then $n_2 \leftarrow n_2 + 1;\ n_1 \leftarrow 0;\ n_3 \leftarrow 1$ else if $n_2 < L$ then $n_2 \leftarrow n_2 + 1;\ n_1 \leftarrow 0$ end if Return Intens $\leftarrow \lambda_1$
e7: if $n_1 = 5$ if $n_3 = 0$ and $n_2 < L$ then $n_2 \leftarrow n_2 + 1;\ n_1 \leftarrow 0;\ n_3 \leftarrow 1$ else if $n_2 < L$ then $n_2 \leftarrow n_2 + 1;\ n_1 \leftarrow 0$ end if Return Intens $\leftarrow \lambda_2$	e8: if $n_3 = 1$ if $n_2 > 1$ then $n_2 \leftarrow n_2 - 1;\ n_3 \leftarrow 1$ else $n_2 \leftarrow 0;\ n_3 \leftarrow 0$ end if end if Return Intens $\leftarrow \mu$

2.2.2. G/M/1 SISTEMOS TYRIMAS, KAI G – KOKSO SKIRSTINYS

Nagrinėkime aptarnavimo sistemos modelį *G/M/1*, kai skirstinio funkcija *G* aproksimuojama Kokso skirstiniu (1.2.2.2 skyrelis). 2.7 lentelėje pateiktas sistemos aprašymo įvykių kalba pseudokodas.

2.7 lentelė

G/M/1 sistemos pseudokodas (G – Kokso skirstinys)

e1: if $n_1 = 0$ then $n_1 \leftarrow 1$ end if Return Intens $\leftarrow C$	e2: if $n_1 = 1$ then $n_1 \leftarrow 2$ end if Return Intens $\leftarrow p \cdot \lambda_1$
e3: if $n_1 = 1$ if $n_3 = 0$ and $n_2 < L$ then $n_2 \leftarrow n_2 + 1;\ n_1 \leftarrow 0;\ n_3 \leftarrow 1$ else if $n_2 < L$ then $n_2 \leftarrow n_2 + 1;\ n_1 \leftarrow 0$ end if Return Intens $\leftarrow (1-p) \cdot \lambda_1$	e4: if $n_1 = 2$ if $n_3 = 0$ and $n_2 < L$ then $n_2 \leftarrow n_2 + 1;\ n_1 \leftarrow 0;\ n_3 \leftarrow 1$ else if $n_2 < L$ then $n_2 \leftarrow n_2 + 1;\ n_1 \leftarrow 0$ end if Return Intens $\leftarrow \lambda_2$
e5: if $n_3 = 1$ if $n_2 > 1$ then $n_2 \leftarrow n_2 - 1;\ n_3 \leftarrow 1$ else $n_2 \leftarrow 0;\ n_3 \leftarrow 0$ end if end if Return Intens $\leftarrow \mu$	

Sumodeliavus sistemą, kurioje maksimalus paraiškų skaičius $L = 3$, gauname $n = 12$ eilės perėjimo tikimybių matricą (tikimybės suapvalintos):

$$P = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.24 & 0.76 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1e-05 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.3 & 0 & 0 & 0 & 0.17 & 0.53 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.65 & 0 & 0 & 0 & 0.35 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1e-05 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.3 & 0 & 0 & 0 & 0.17 & 0.53 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.65 & 0 & 0 & 0 & 0.35 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1e-05 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.65 & 0 & 0 & 0 & 0.35 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{pmatrix} \qquad \textbf{(2.4)}$$

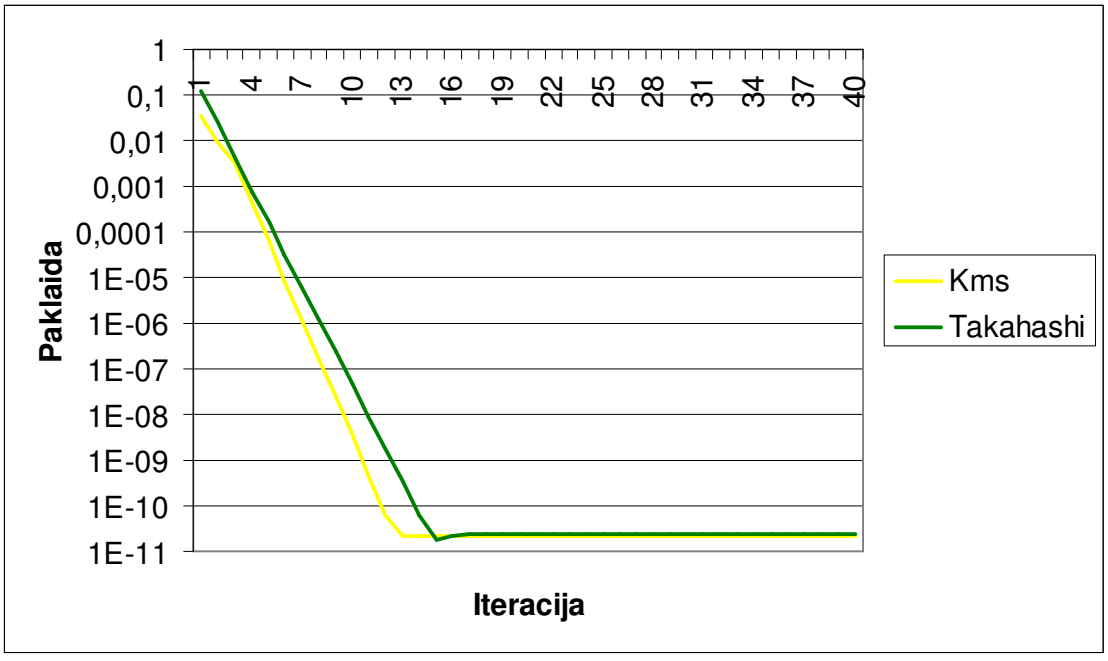
Atkreipkime dėmesį į tai, kad modelių *G/M/1* (nesvarbu ar *G* yra Erlango mišinys, ar Kokso skirstinys) perėjimo tikimybių matrica P turi panašią formą. Tiek (2.3), tiek (2.4) matricų elementai išsidėstę virš bei po pagrindine įstrižaine. Todėl taikydami panašią sprendimo schemą kaip (2.3) atveju, pradinė matrica išskaidome į submatricas. Tik šį kartą gaunami nebe 2, bet 3 submodeliai (raudona punktyrinė linija). Pritaikius *IAD* algoritmus, gauname sprendinius:

2.8 lentelė

G/M/1 sistemos sprendinys (G – Kokso skirstinys)

Įprastas būdas	IAD algoritmas		Paklaida	
	Kms	Takahashi	Kms	Takahashi
0,00000086868746	0,00000086868746	0,00000086868746	0,0000E+00	0,0000E+00
0,04233824364464	0,04233824364382	0,04233824364382	8,2000E-13	8,2000E-13
0,05462552782781	0,05462552782216	0,05462552782216	5,6500E-12	5,6500E-12
0,08686961450101	0,08686961449474	0,08686961449474	6,2700E-12	6,2700E-12
0,13914409885030	0,13914409884761	0,13914409884761	2,6900E-12	2,6900E-12
0,06819043078140	0,06819043077306	0,06819043077306	8,3400E-12	8,3400E-12
0,09738529305576	0,09738529305144	0,09738529305144	4,3200E-12	4,3200E-12
0,17180580749150	0,17180580750323	0,17180580750323	1,1730E-11	1,1730E-11
0,06907464622318	0,06907464619380	0,06907464621109	2,9380E-11	1,2090E-11
0,11499641904135	0,11499641903737	0,11499641906616	3,9800E-12	2,4810E-11
0,11499526907716	0,11499526907318	0,11499526910197	3,9800E-12	2,4810E-11
0,04057378081842	0,04057378078709	0,04057378079725	3,1330E-11	2,1170E-11

Tuo pačiu pridėkime algoritmų konvergavimo greičio grafiką:



2.4 pav. IAD algoritmų palyginimas konvergavimo greičio prasme G/M/1 modelyje

Sėkmingas blokų parinkimas mažina iteracijų skaičių.

2.3. BLOKINĖS STRUKTŪROS PAIEŠKOS MODIFIKACIJA

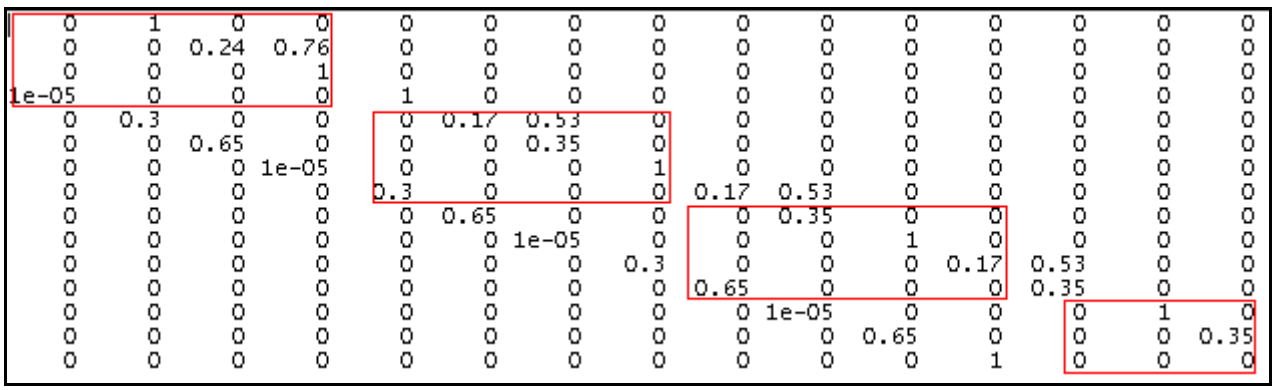
Iš teorinės dalies prisimename, kad perėjimo tikimybių matrica P skaidoma į submatricas remiantis 1.1 lentelėje pateikta paieškos procedūra. Didėjant dekompozicijos parametrui dalis perėjimo tikimybių panaikinamos jas pakeičiant nuliais. Tačiau tyrimų eigoje pastebėjome, jog nagrinėjamų aptarnavimo sistemų modeliuose didžioji dalis tikimybių yra dešimtųjų eilės. Taigi dekompozicijos parametras iš esmės nekeičia matricos P struktūros. Be to, modeliuojamų sistemų perėjimo tikimybių matricos pasižymi tuo, kad pagrindinėje įstrižainėje visi elementai lygūs nuliui. Vadinasi, kad ir kaip beskaidytume matricą P į submatricas, kiekvienoje iš jų pagrindinėje įstrižainėje liks tie patys nuliai. O tai reiškia, kad sistema kiekvienu perėjimu pakeis būseną, t.y. niekada neliks toje pačioje. Darydami tokią prielaidą, galime ieškoti tokios submatricos, kurioje visose eilutėse ir stulpeliuose būtų bent po vieną nenulinį elementą (neatsižvelgiant į jo dydį, t.y. tikimybės pereiti į kitą būseną dydį). Tokiu būdu papildykime

2.9 lentelė

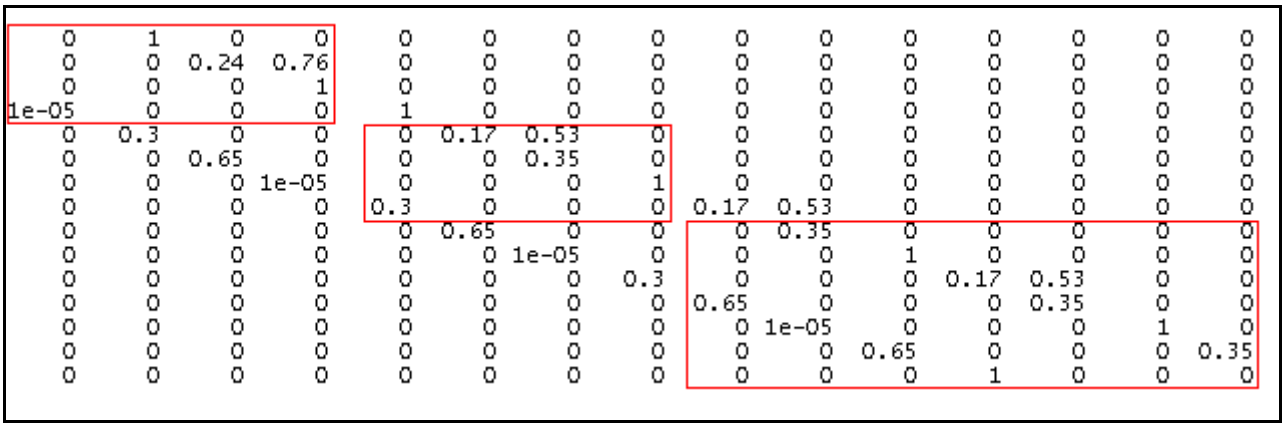
Blokinės struktūros paieškos modifikacija

Etapas	Veiksmas
1.	Įvedamas dekompozicijos parametras γ, kuris nuo 10^{-10} iki 10^{-1}
2.	Visi matricos P elementai, mažesni ar lygūs parametrui γ, pakeičiami nuliais
3.	Naujoji matrica traktuojama kaip orientuoto grafo vaizdavimas ir vykdoma stipriai sujungtų grafo viršūnių paieška [2]
4.	Kitu atveju, bloką prilyginti submatricai, kurioje kiekvienoje eilutėje ir stulpelyje yra bent vienas nenulinis elementas

Pažiūrėkime, kaip siūloma blokų paieškos alternatyva veikia analizuojamoje sistemoje *G/M/1*, kai *G* – Kokso skirstinys. 2.5 paveiksle matome, kad aptartu būdu gavome 4-ių blokų struktūrą. Tiesa paskutinė 3×3 dydžio submatrica netenkina užsibrėžtų sąlygų (liko kaip metodo liekana), todėl paskutiniuosius 2 blokus galime apjungti į vieną (2.6 pav.).



2.5 pav. Blokų paieška (1)



2.6 pav. Blokų paieška (2)

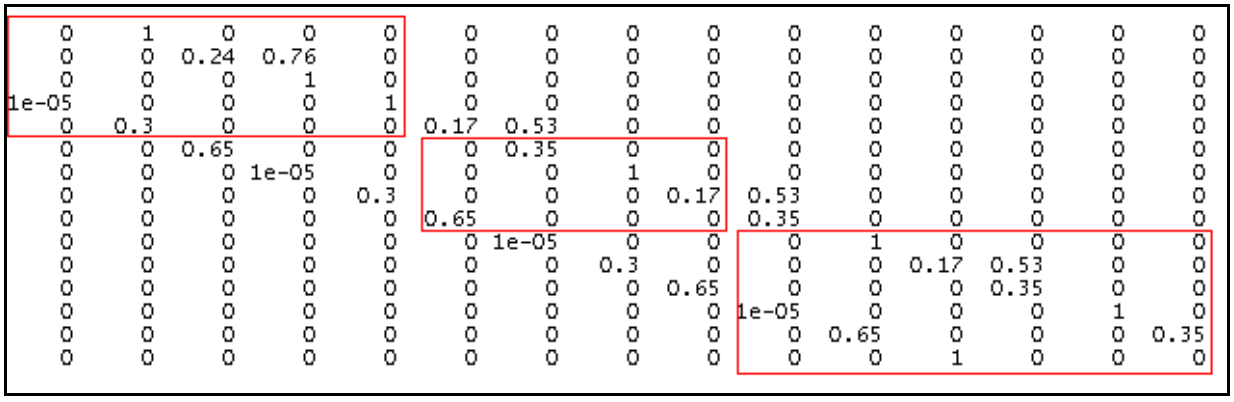
Pritaikius *IAD* algoritmus, gauname stacionarių tikimybių vektorių:

2.10 lentelė

G/M/1 sistemos sprendinys (G – Kokso skirstinys)

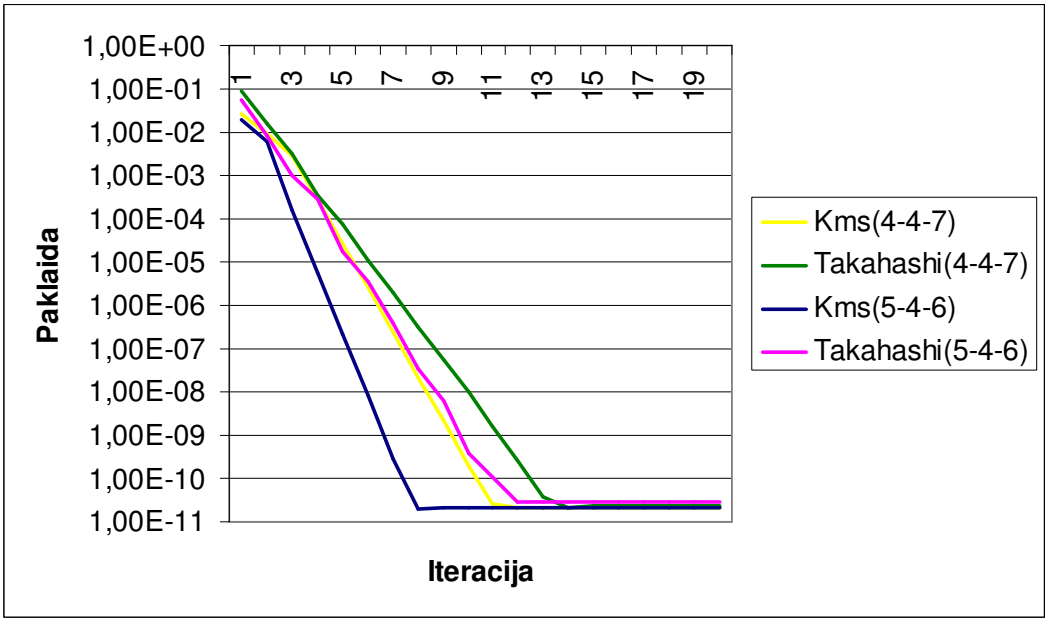
Įprastas būdas	IAD algoritmas		Paklaida	
	Kms	Takahashi	Kms	Takahashi
0,00000063061919	0,00000063061919	0,00000063061919	0,0000E+00	0,0000E+00
0,02943952541480	0,02943952541430	0,02943952541430	5,0000E-13	5,0000E-13
0,04064177556118	0,04064177555803	0,04064177555803	3,1500E-12	3,1500E-12
0,06306254957512	0,06306254957158	0,06306254957158	3,5400E-12	3,5400E-12
0,09675285617534	0,09675285617369	0,09675285617369	1,6500E-12	1,6500E-12
0,05148642240037	0,05148642239572	0,05148642239572	4,6500E-12	4,6500E-12
0,06912878481185	0,06912878480936	0,06912878480936	2,4900E-12	2,4900E-12
0,11072747215692	0,11072747216313	0,11072747216313	6,2100E-12	6,2100E-12
0,05426430417454	0,05426430414662	0,05426430416784	2,7920E-11	6,7000E-12
0,07749712469430	0,07749712468790	0,07749712471820	6,4000E-12	2,3900E-11
0,13671908289349	0,13671908286861	0,13671908292206	2,4880E-11	2,8570E-11
0,05496821140868	0,05496821136435	0,05496821138584	4,4330E-11	2,2840E-11
0,09151194116445	0,09151194113589	0,09151194117167	2,8560E-11	7,2200E-12
0,09151102604504	0,09151102601648	0,09151102605226	2,8560E-11	7,2200E-12
0,03228829290472	0,03228829286452	0,03228829287715	4,0200E-11	2,7570E-11

Jeigu į blokų paieškos algoritmą įtrauktumėme sąlygą, kad įstrižainiuose blokuose turi būti tikimybės arčiau vieneto, o jų ribų liktų artimi nuliui elementai (1.3.1 skyrelis), tuomet automatiškai keičiasi blokinė struktūra (2.7 paveikslas).



2.7 pav. Blokų paieška (3)

Blokinė $(4\times4,\ 4\times4$ ir $7\times7)$ struktūra pasikeičia į $(5\times5,\ 4\times4$ ir $6\times6)$. Tačiau stacionarių tikimybių skaičiavime didelio pokyčio nepastebėjome. Tikslumas išliko toks pat. Tiesa, iteracijų skaičius, kuriuo pasiekiamas maksimalus tikslumas, išsiskyrė.



2.8 pav. Konvergavimo pokytis, kai blokinė struktūra skirtinga

Kms algoritmas, kuomet matrica P išskaidyta į $(5\times5,\ 4\times4$ ir $6\times6)$ dydžio submatricas, konvergavo greičiausiai. Netgi atvejis su (4-4-7) išskaidymu *KMS* algoritmui leido aplenkti *Takahashi* algoritmą.

2.4. DIDELĖS APIMTIES APTARNAVIMO SISTEMŲ MODELIAI

Iki šiol tyrinėjome aptarnavimo sistemas, kurias apibūdina mažo matavimo perėjimo tikimybių iš būsenos į būseną matrica P. Pagrindinis tikslas buvo įsitikinti, ar dekompozicijos metodas tinka problemos sprendimui.

Analizuojami aptarnavimo sistemų modeliai generuoja didelio matavimo matricas P, programos vartotojas nurodo didesnį maksimalų paraiškų skaičių L. Pavyzdžiui, kai $L=30$, tai *G/M/1* modelyje su Erlango mišinio aproksimacija turime nagrinėti matricą P, kurios dydis 124×124. Todėl svarbu sužinoti, kaip *IAD* algoritmai elgiasi tokiose situacijose. Tikėkimės, kad *KMS* ir *Takahashi* algoritmai gerai susidoros su savo užduotimis.

Sakykime nagrinėjame *G/M/1* modelį, kai G – Kokso skirstinys. Kadangi gauti sprendiniai užima daug vietos, pateiksime vidutines paklaidas kiekvienam metodui. 2.11 lentelėje pavaizduoti skaičiavimo rezultatai, kai maksimalus paraiškų skaičius kinta nuo 40 iki 160. Kartu kinta ir perėjimo tikimybių matricos dydis – nuo 123 iki 483.

2.11 lentelė

IAD algoritmų paklaidos, kai L = 40,80,160

Maksimalus paraiškų skaičius L	IAD algoritmo paklaida	
	Kms	Takahashi
40	1,4055E-12	1,6567E-12
80	3,5846E-011	9,6571E-012
160	3.5866E-011	7.9955E-011

Iš lentelės matyti, kad algoritmų tikslumas išliko toks pat kaip ir nedidelės apimties modeliuose. Tai galima būtų paaiškinti teisingai parinkta matricos dalijimo strategija. Kadangi tyrimų metu pastebėta, jog matricos P elementai kartojasi, tai buvo galima panašiai elgtis ir su blokais – mažesnių matricų dalijimą atkartoti didelio matavimo matricose. *G/M/1* (*G* – Kokso skirstinys) modelio atveju standartinis skaidymo algoritmas (2.9 lentelė) suskaido į 4×4 dydžio submatricas (paskutinio bloko dydis skiriasi priklausomai nuo pradinės matricos eilės). Tačiau pradedant matricos $p_{5,6}$ elementu išryškėja tam tikras dėsningumas (2.9 pav.). Tikimybės pereiti iš 5 būsenos į 6, iš 8 į 9, iš 11 į 12 ir t.t. yra vienodos (paryškinta žalia spalva).

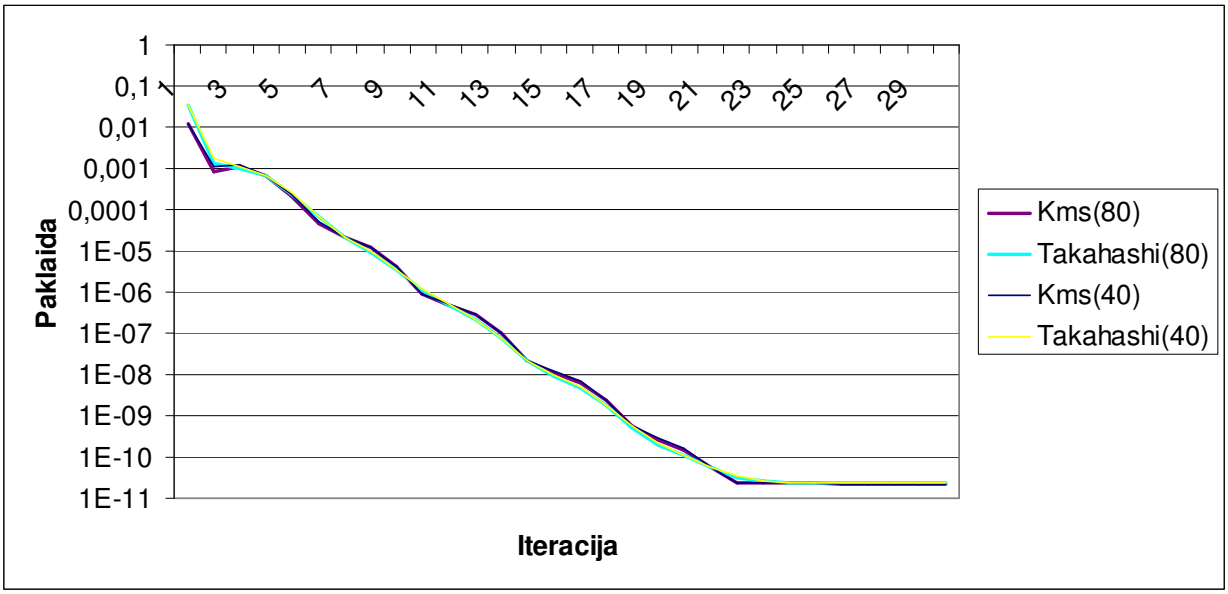
	Būsena6	Būsena7	Būsena8	Būsena9	Būsena10	Būsena11	Būsena12	Būsena13	Būsena14	Būsena15
Būsena5	0,165887	0,529844	0	0	0	0	0	0	0	0
Būsena6	0	0,346967	0	0	0	0	0	0	0	0
Būsena7	0	0	0,99999	0	0	0	0	0	0	0
Būsena8	0	0	0	0,165887	0,529844	0	0	0	0	0
Būsena9	0,653033	0	0	0	0,346967	0	0	0	0	0
Būsena10	0	9,9999E-6	0	0	0	0,99999	0	0	0	0
Būsena11	0	0	0,304269	0	0	0	0,165887	0,529844	0	0
Būsena12	0	0	0	0,653033	0	0	0	0,346967	0	0
Būsena13	0	0	0	0	9,9999E-6	0	0	0	0,99999	0
Būsena14	0	0	0	0	0	0,304269	0	0	0	0,165887
Būsena15	0	0	0	0	0	0	0,653033	0	0	0
Būsena16	0	0	0	0	0	0	0	9,9999E-6	0	0

2.9 pav. Optimalaus matricos skaidymo parinkimas G/M/1 modelyje (G – Kokso skirstinys)

Tikimybių pasikartojimas matricoje suteikia galimybę parinkti optimalų matricos skaidymą į submatricas. Tokiu būdu pirmasis blokas yra pirmų penkių būsenų perėjimo tikimybės, t.y. P_{11} yra 5×5 dydžio submatrica. Tolimesnį skaidymą susiekime su pastebėtu elementų pasikartojimu. Taikant įprastą skaidymo algoritmą, vėlgi gauname 4×4 dydžio submatricas. Tačiau atsižvelgus į metodikos reikalavimą, kad blokuose turi būti artimos vienetui tikimybės, naudinga submatricos dydį praplėsti iki 6×6 (2.9 pav. paryškinta raudonai). Tokiu atveju užtikriname, jog $p_{10,11}$ elementas patenka į bloką. Dėl matricos P struktūros pasikartojimo submatricos P_{22}, P_{33} ir t.t. apjungia tuos pačius elementus.

Paskutinis bloko dydis priklauso nuo situacijos. Jei skaidymas baigiasi mažesnės nei 6-tos eilės submatrica, tai toks dydis ir paliekamas. Vadinasi, *G/M/1* (G – Kokso skirstinys) modelio perėjimo tikimybių matricos blokinė struktūra atrodytų taip: P_{11} (5 eilės), P_{22} (6 eilės), P_{33} (6 eilės),..., P_{NN} (mažesnės arba lygi 6-tos eilės submatricai).

Naudodamiesi anksčiau aptarta skaidymo strategija, apskaičiuotos apytikslės stacionarių tikimybių reikšmės (2.11 lentelė). Algoritmų konvergavimas pateiktas 2.10 paveiksle.



2.10 pav. IAD algoritmų konvergavimo greičio palyginimas

Apjungus abiejų algoritmų (kai L = 40,80) konvergavimo grafikus į vieną, sunku įžiūrėti didesnius skirtumus – visi grafikai persidengia vienas ant kito. Dėl šios priežasties atsisakėme grafiko, kai L = 160, nes toliau nagrinėjant didesnes sistemas, stacionarios tikimybės vis mažėja ir nedaro esminių pokyčių skaičiavimuose.

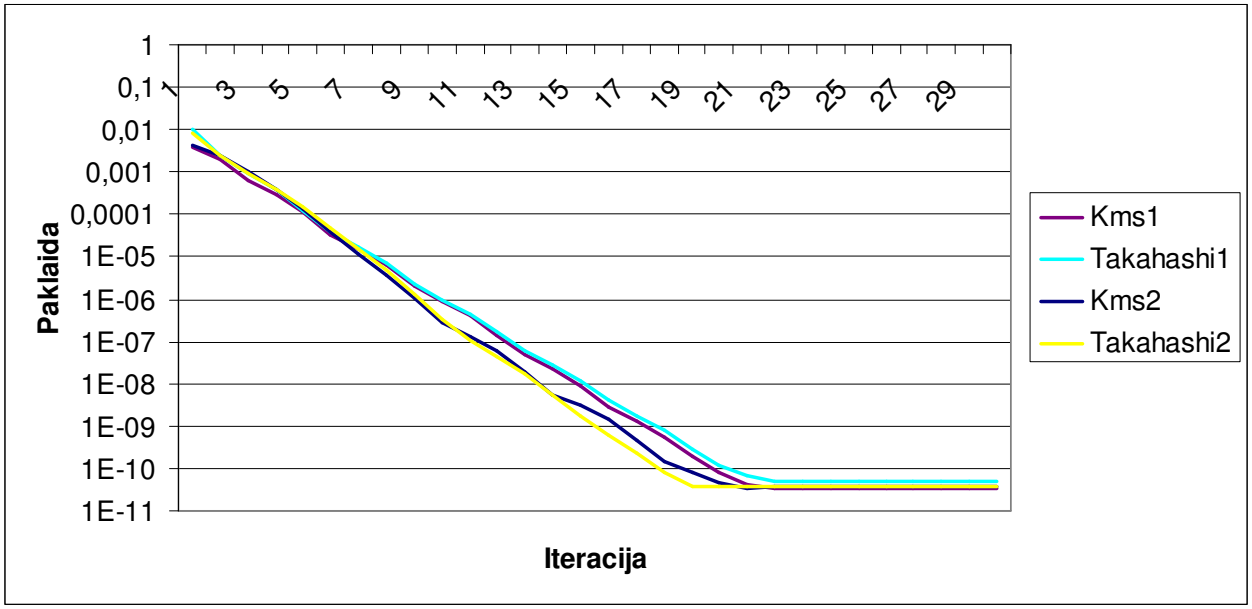
Analogišką matricos P skaidymo strategiją galime pritaikyti, kai G yra Erlango mišinys. Dėl panašios matricos struktūros vėl pastebime elementų kartojimosi dėsningumą (2.11 pav.).

	Būsena6	Būsena7	Būsena8	Būsena9	Būsena10	Būsena11	Būsena12	Būsena13	Būsena14	Būsena15
Būsena6	0	0,310607	0,689383	0	0	0	0	0	0	0
Būsena7	0	0	0	0,346967	0	0	0	0	0	0
Būsena8	0	0	0	0,695731	0	0	0	0	0	0
Būsena9	0	0	0	0	0,99999	0	0	0	0	0
Būsena10	9,9999E-6	0	0	0	0	0,310607	0,689383	0	0	0
Būsena11	0	0,653033	0	0	0	0	0	0,346967	0	0
Būsena12	0	0	0,304269	0	0	0	0	0,695731	0	0
Būsena13	0	0	0	9,9999E-6	0	0	0	0	0,99999	0
Būsena14	0	0	0	0	9,9999E-6	0	0	0	0	0,310607
Būsena15	0	0	0	0	0	0,653033	0	0	0	0
Būsena16	0	0	0	0	0	0	0,304269	0	0	0
Būsena17	0	0	0	0	0	0	0	9,9999E-6	0	0

2.11 pav. Optimalaus matricos skaidymo parinkimas G/M/1 modelyje (G – Erlango mišinys)

Vadovaudamiesi ta pačia logika, galime parinkti perėjimo tikimybių matricos skaidymo būdą. Kadangi šį kartą kartojimosi elementas pasistūmėjo viena pozicija ($p_{6,7}$ elementas), tai pirmasis skaidymo blokas P_{11} yra 6-tos eilės. Tolimesnis skaidymas vyksta apjungiant 8-ių būsenų perėjimo tikimybes į vieną submatricą (2.11 pav. paryškinta raudonai). Tokiu būdu gauname skaidymo strategiją: P_{11} (6×6), P_{22} (8×8), P_{33} (8×8),..., P_{NN} (mažesnės arba lygi 8-tos eilės submatricai).

Įprastas skaidymo algoritmas (2.9 lentelė) pateikia tokį skaidymą: submatricos, kurių dydžiai 5×5, 5×5, 6×6, kartojasi tol, kol padalinama visa matrica P. Palyginkime abiejų skaidymo strategijų konvergavimą. Tam tikslui sumodeliuokime sistemą, kurioje maksimalus paraiškų skaičius $L=25$, o perėjimo tikimybių matricos dydis $n=104$ (2.12 paveiksle pirmuoju numeriu pažymėtas įprastas skaidymas, antruoju – parinktasis).



2.12 pav. Skaidymo strategijų palyginimas G/M/1 modelyje (G – Erlango mišinys)

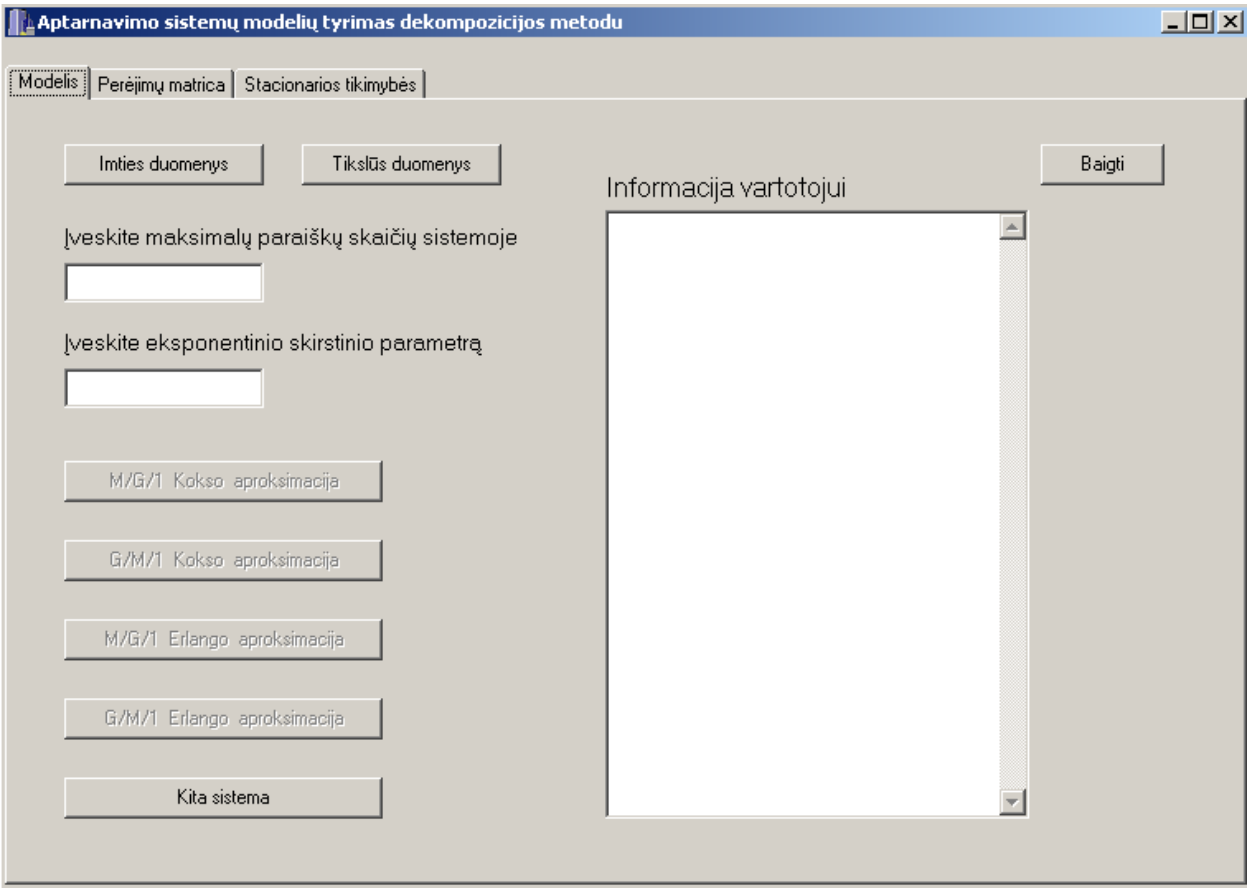
Matome, kad parinkta skaidymo strategija įgyja nežymų pranašumą konvergavimo greičio prasme. Pakartojus bandymus su kitokio dydžio matricomis, tyrimo rezultatai išlieka panašūs. Vadinasi, galime teigti, kad nauja skaidymo strategija šiuo atžvilgiu yra optimalesnė.

M/G/1 modelių atveju nepavyko surasti naujos skaidymo strategijos. Jau pradinis skaidymas nedavė jokių rezultatų, t.y. algoritmas nerasdavo reikalavimus tenkinančių blokų. Todėl *M/G/1* modeliams dekompozicijos metodas netinka – po skaidymo vis tiek turime tik pradinę matricą P.

3. PROGRAMINĖ REALIZACIJA

Magistro baigiamajame darbe atitinkamais algoritmais išplėtėme universalią programinę įrangą, skirtą modeliuoti aptarnavimo sistemas. Programa parašyta „C++ Builder“ programavimo kalba. Paprastų aptarnavimo sistemų M/G/1 ir G/M/1 modeliavimas automatizuotas – t.y., vartotojas gali perduoti programai imties duomenis (gautus stebint atsitiktinį dydį, turintį skirstinį G) ar skirstinio G pradinius momentus. Sudėtingesnių sistemų aprašymą vartotojas gali pateikti pats. Sistema aprašoma įvykių kalba, kurios pavyzdžių galima rasti [7]. Sistemos veikimo principai aprašyti [7]. Įvedus parametrus programa modeliuoja būsenų aibę kartu su perėjimų matrica. Vėliau vartotojas pasirenka, kokiu metodu skaičiuoti stacionarias tikimybes. Statistinių charakteristikų skaičiavimo metodai gali skirtis įvairioms sistemoms. Bendru atveju, kaip ir sistemos aprašymu įvykių kalba, jų sukūrimu turi rūpintis vartotojas.

Paleidus programą vartotojas mato programos langą (3.1 pav.), kuriame yra modeliuojama vartotojo nurodyta aptarnavimo sistema.



3.1 pav. Programos lango puslapis „Modelis“

Puslapio „Modelis“ mygtukų paaiškinimai:

„Imties duomenys“ – nuskaitomi duomenys, gauti stebint tam tikrą atsitiktinį dydį. Duomenų failas „Imties duomenys.txt“ saugomas programos kataloge „Duomenys“. Duomenys – realūs teigiami skaičiai. Sėkmingai nuskaičiusi duomenis programa apskaičiuoja empirinius pradinius momentus ir įvertina galimą skirstinių aproksimavimą.

„Tikslūs duomenys“ – nuskaitomi aproksimuojamo skirstinio 3 pradiniai momentai . Duomenų failas „Tikslus duomenys“ saugomas programos kataloge „Duomenys“. Sėkmingai nuskaičiusi duomenis programa įvertina galimą skirstinių aproksimavimą.

„*M/G/1* Kokso aproksimacija“ – jei nuskaityti duomenys tenkina aproksimavimo sąlygas, programa apskaičiuoja ir išveda į ekraną sistemos *M/G/1* būsenų perėjimo tikimybių matricą, kai skirstinys *G* aproksimuotas Kokso skirstiniu.

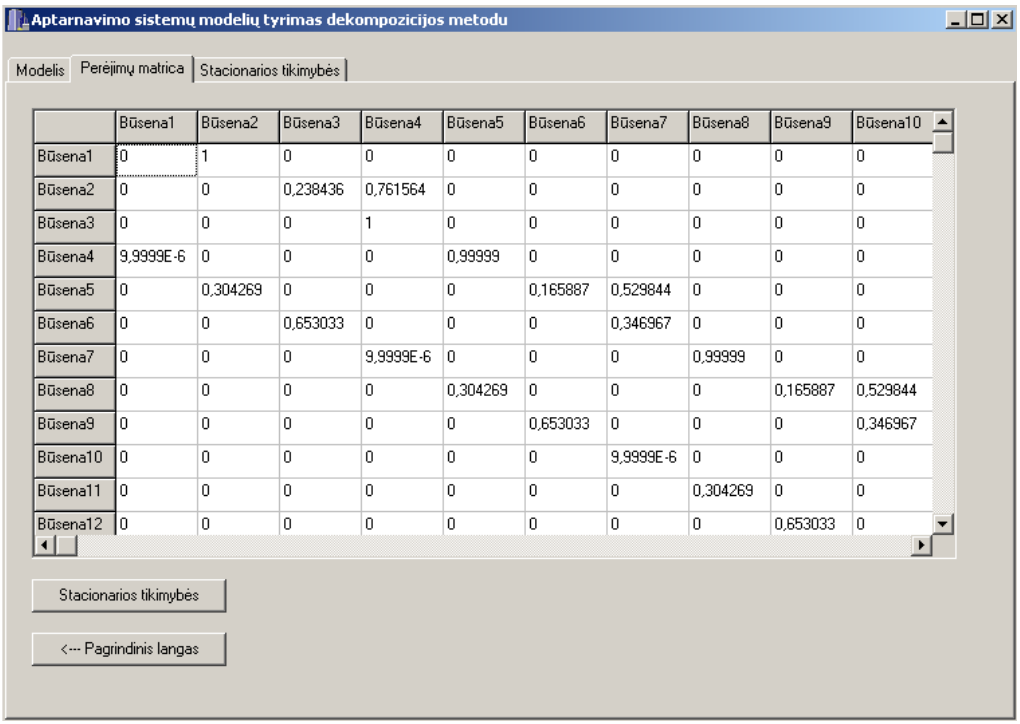
„*G/M/1* Kokso aproksimacija“ – jei nuskaityti duomenys tenkina aproksimavimo sąlygas, programa apskaičiuoja ir išveda į ekraną sistemos *G/M/1* būsenų perėjimo tikimybių matricą, kai skirstinys *G* aproksimuotas Kokso skirstiniu.

M/G/1 Erlango aproksimacija“ – programa apskaičiuoja ir išveda į ekraną sistemos *M/G/1* būsenų perėjimo tikimybių matricą, kai skirstinys *G* aproksimuotas Erlango mišiniu.

„*G/M/1* Erlango aproksimacija“ – programa apskaičiuoja ir išveda į ekraną sistemos *G/M/1* būsenų perėjimo tikimybių matricą, kai skirstinys *G* aproksimuotas Erlango mišiniu.

„Kita sistema“ – apskaičiuoja vartotojo aprašytos sistemos charakteristikas. Šiuo atveju, vartotojas pats aprašo reikiamą sistemą, nurodo duomenų failą.

„Baigti“ – baigia programos darbą ir uždaro programos langą.



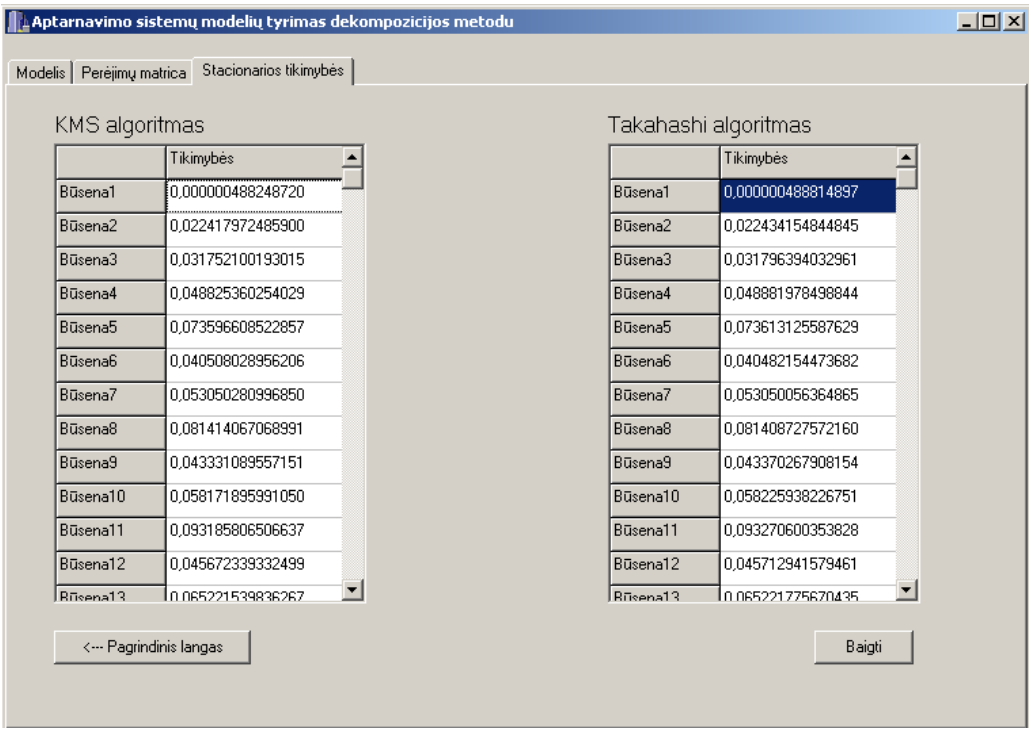
	Būsena1	Būsena2	Būsena3	Būsena4	Būsena5	Būsena6	Būsena7	Būsena8	Būsena9	Būsena10
Būsena1	0	1	0	0	0	0	0	0	0	0
Būsena2	0	0	0,238436	0,761564	0	0	0	0	0	0
Būsena3	0	0	0	1	0	0	0	0	0	0
Būsena4	9,9999E-6	0	0	0	0,99999	0	0	0	0	0
Būsena5	0	0,304269	0	0	0	0,165887	0,529844	0	0	0
Būsena6	0	0	0,653033	0	0	0	0,346967	0	0	0
Būsena7	0	0	0	9,9999E-6	0	0	0	0,99999	0	0
Būsena8	0	0	0	0	0,304269	0	0	0	0,165887	0,529844
Būsena9	0	0	0	0	0	0,653033	0	0	0	0,346967
Būsena10	0	0	0	0	0	0	9,9999E-6	0	0	0
Būsena11	0	0	0	0	0	0	0	0,304269	0	0
Būsena12	0	0	0	0	0	0	0	0	0,653033	0

3.2 pav. Programos lango puslapis „Perėjimų matrica“

Puslapio „Perėjimų matrica“ mygtukų paaiškinimai:

„Stacionarios tikimybės“ – atlieka matricos skaidymą ir apskaičiuoja vartotojo nurodytos aptarnavimo sistemos stacionarias tikimybes, panaudojant *KMS* bei *Takahashi* algoritmus. Skaidymo rezultatai išsaugojami faile „Skaidymas“, kuris saugomas programos kataloge „Rezultatai“. Tame pačiame kataloge failuose „KMS“ ir „Takahashi“ atitinkamai saugomi skaičiavimo rezultatai.

„Pagrindinis langas“ – grįžta į puslapį „Modelis“, kad vartotojas galėtų modeliuoti kitą sistemą.



3.3 pav. Programos lango puslapis „Stacionarios tikimybės“

Paskutiniame programos lango puslapyje „Stacionarios tikimybės“ išvedamos *KMS* ir *Takahashi* algoritmais apskaičiuotos stacionarios tikimybės.

Imties duomenų faile „Imties duomenys.txt“ duomenys – realūs teigiami skaičiai, surašyti po vieną eilutėje, be jokių papildomų skyriklių.

Duomenų faile „Tikslus duomenys“ turi būti trys teigiami skaičiai – konkretaus skirstinio pradiniai momentai. Skaičiai surašyti po vieną eilutėje, be jokių papildomų skyriklių.

Sistemų *M/G/1* ir *G/M/1* duomenų failai saugomi programos kataloge “Duomenys” ir vartotojas jų neturėtų koreguoti. Tuo tarpu modeliuojant kitokio tipo sistemas vartotojas privalo pats sukurti reikiamus duomenų failus. Pateikiame šių failų struktūrą:

1) Pirmoje duomenų failo eilutėje turi būti įvesta pradinė sistemos būsena. Ji turi būti rašoma po „=“ ir „ “. gale turi būti „;“ simbolis. Pvz.

pradine busena= 0 0 0;

2) Antroje duomenų failo eilutėje turi būti aptarnaujančių įrenginių intensyvumai. Pradžioje eilutės padedami „=“ ir „ “, o gale – „;“. Pvz.

intensyvumai= 0.75 1 2

3) Trečioje eilutėje įvedamas paraiškų srautų skaičius. Pradžioje eilutės padedami „=“ ir „ “, o gale – „;“. Pvz.

srautu skaicius= 1;

4) Ketvirtoje eilutėje įvedamas sistemos įvykių skaičius. Pradžioje eilutės padedami „=“ ir „ “, o gale – „;“. Pvz.

ivykiu skaicius= 4;

Jei vartotojas nori modeliuoti savo pasirinktą (ne *M/G/1* ar *G/M/1*) aptarnavimo sistemą, jos aprašymą reikia pateikti formoje *TForm1* esančiame metode *Busenos(int *, int)*. Sistema modeliavimui, turi būti pateikiama *C++ Builder* kalboje naudojamu *switch-case* sakiniu 3.2 pav .

```
switch (poz)
 {
  case 0:           //tikrinam ivyki e{1}
  {
   if(xx[0]+xx[1]<L)
   {
    xx[1]++;
    radom=true;
   }
   else
   {
    xx[0]++;
    radom=true;
   }
   return Intens0;
  }    //ivykio e{1} aprasymo pabaiga
...
...
  case n:           //tikrinam ivyki e{n+1}
  {
   if(...)
   {
    ...;
    radom=true;
   }
   else
   {
    ...;
    radom=true;
   }
   return ...;
  }
 }   //ivykio e{n+1} aprasymo pabaiga
}   //metodo pabaiga
```

3.4 pav. Sistemos aprašymas *Switch-Case* sakiniu

Čia `xx[i]` yra tiriamos būsenos vektorius kur `xx[i]` yra (i +1) -oji vektoriaus koordinatė. `Intens0` yra perėjimo intensyvumas iš atitinkamos būsenos. Modeliuojant aptarnavimo sistemas, intensyvumai gali būti įvedami pradinių duomenų byloje, arba nurodomi tiesiogiai aprašant sistemą. Atlikus būsenos koordinatės keitimą, reikalinga pažymėti, kad nauja būsena rasta, sakiniu „`radom=true;`”. *Case* sakinio pabaigoje reikia gražinti reikiamą perėjimo į naujai rastą būseną intensyvumą, sakiniu „`return Intens;`”

IŠVADOS

- Bendru atveju dekompozicijos schema yra sunkiai įgyvendinamas uždavinys analizuojamiems aptarnavimo sistemų modeliams. Dėl savitos perėjimo tikimybių matricos P struktūros, kai *G/M/1* ir *M/G/1* (*G* aproksimuojamas Erlango mišiniu ar Kokso skirstiniu) modeliuose sunku matricą P suskaidyti į submatricas.
- Papildomos prielaidos, atsiradusios tyrimų eigoje, bei skaidymo į submatricas modifikacija suteikė daugiau galimybių aptarnavimo sistemų stacionarių tikimybių skaičiavimui. G/M/1 modelių atveju surasta optimali skaidymo strategija, tačiau M/G/1 modeliams dekompozicijos schemos naudoti nepatartina.
- *IAD* algoritmų panaudojimas stacionarių tikimybių skaičiavime duoda efektyvius rezultatus. Metodų tikslumas svyruoja tarp 10^{-12} ir 10^{-10} skaitmenų po kablelio. Iteracijų skaičius neviršija 30 priklausomai nuo matricos P eilės.
- Lyginant abu *IAD* algoritmus tarpusavyje, *KMS* algoritmą būtų galima išskirti dėl konvergavimo greičio, nors abu metodai tikslumu vienas kitam nenusileidžia.
- Didėjant matricos P matavimui, *KMS* ir *Takahash*i algoritmai išlieka ne mažiau efektyvūs nei analizuojant paprastas aptarnavimo sistemas.

PADĖKOS

Norėčiau padėkoti darbo vadovui doc. dr. E. Valakevičiui už pasiūlytą magistro baigiamojo darbo temą, pagalbą ir konsultacijas atliekant darbe iškeltas užduotis.

Taip pat didelis ačiū keliauja doktorantui Mindaugui Šnipui už pagalbą, gilinantis į jo sukurtą programinę įrangą, bei sugaištą laiką patarimams su darbo tematika susijusiais klausimais.

LITERATŪROS SĄRAŠAS

1. Adan, I.; Resing, J. Queueing Theory [interaktyvus]. TUE, Department of Mathematics and Computing Science, 2002. – [žiūrėta 2009-02-11]. Prieiga per internetą: <http://www.win.tue.nl/~iadan/queueing.pdf>.
2. Gross, J. L.; Yellen, J. Graph theory and its applications. CRC Press, Boca Raton, Florida, 1999. 585 p.
3. I. Ipsen and C.D. Meyer. Uniform stability of Markov chains. Department of Mathematics. North Carolina State University, Raleigh, 1992.
4. Mickevičius, G., Valakevičius, E., Modelling of non-Markovian queuing systems, Technological and economic development of economy, Vol XII, No 4, 2006, p. 295-300.
5. Plukas K. Skaitiniai metodai ir algoritmai. Kaunas, 2001. 548 p.
6. Pranevičius, H., Pranevičienė, I. Masinio aptarnavimo teorijos elementai. Vilnius, 1980.
7. Pranevičius, H., Valakevičius, E. Numerical Models of Systems Specified by Marcovian processes. Kaunas, 1996.
8. H. A. Simon and Ando. Aggregation of variables in dynamic systems. Econometrica, 29:111-138, 1961.
9. Stewart, W. J. Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton, New Jersey, 1994. 539 p.
10. Šnipas, M. Stochastinių sistemų aproksimavimas Markovo modeliais. Magistro darbas. KTU, Fundamentaliųjų mokslų fakultetas, 2008.

PRIEDAI

1 PRIEDAS. PROGRAMOS TEKSTAS

```
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <Dialogs.hpp>
#include <Grids.hpp>
#include <algorithm>
#include <iostream>
#include <functional>
#include <fstream>
using namespace std;
const int itmax1 = 5;
const int itmax2 = 50;
struct st
{
float BInt;
int eil;
int st;
};
struct TBusIntens    //uzkoduota perejimu matrica
{
 st BI;
 TBusIntens *kitas;
};

struct TBus
{
 int *sk;
 TBus *kitas;
};
struct TBusA    //~=TBus tik skirtas realiems skaiciams
{
 double *sk;
 TBusA *kitas;
};
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
        TPageControl *PageControl1;
        TTabSheet *TabSheet1;
        TTabSheet *TabSheet2;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TButton *Button4;
        TButton *Button5;
        TButton *Button6;
        TButton *Button7;
        TEdit *Edit1;
        TMemo *Memo1;
        TOpenDialog *OpenDialog1;
        TStringGrid *StringGrid1;
        TButton *Button8;
        TButton *Button9;
        TEdit *Edit2;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TTabSheet *TabSheet3;
        TButton *Button10;
        TStringGrid *StringGrid2;
        TStringGrid *StringGrid3;
        TLabel *Label5;
        TLabel *Label6;
        TButton *Button11;
        TButton *Button12;
        void __fastcall FormCreate(TObject *Sender);
```

```
        void __fastcall Button7Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button8Click(TObject *Sender);
        void __fastcall Button9Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
        void __fastcall Button6Click(TObject *Sender);
        void __fastcall Button12Click(TObject *Sender);
        void __fastcall Button11Click(TObject *Sender);
        void __fastcall Button10Click(TObject *Sender);
        //void __fastcall PageControl1Change(TObject *Sender);
private:    // User declarations
        float *Tintens;
        TBusIntens *TPr; //perejimu tikimybiu koduotas dinaminis sarasas
        TBus *TBPr;
        float *r;      //normuotos stac tikimybes
        float *r_apj;  //apjungtos stac tikimybes, nes vienai busenai dvi fazes
        float m1,m2,m3,c,c2,asim; // momentai
        float kok2_miu1,kok2_miu2,kok2_p;
        float kok3_miu1,kok3_miu2,kok3_p;
        float erl2_miu,erl2_p;
        float erl3_miu1,erl3_miu2,erl3_p;
        int erl2_n,erl3_n;
        float *y;
        int n;
        int L;        //max eiles ilgis
        int M;        //ivykiu skaicius
        int R;          //srautu skaicius
        int K;       //kasu skaicius
        int tipas;
        float lamda;
        bool radom;     //ar rasta nauja busena;
        int IK;         //intensyvumu skaicius
        int N;        //intervalu skaicius  (busenu skaicius)
        TBusA *TBAPr;           //nereik?
        double **P;
        int nn;
        int *ni;
        int nb;
        int *bl;
        double *PI;
        //---------------------------------------------------
        void SuskaidytiIntervalais(double);
        bool ErgodiskumoPatikrinimas();
        void Normavimas();
        bool PatikrinimasVienatiskumoA(double [], int, double);      //~=PatikrinimasVienatiskumo tik
operuosime realiaisiais skaiciais
        void SurastiFaziuIntens();
        void PerejimuTikimybes();
        void IvestiPradSalygas();
        void IsvestiMemo();
        void IsvestiStrGrid();
        void BusenuRadimas(int tipas);
        void Isvedimas(int tipas);
        void IvestiPradSalygasKoksoMG1();
        void IvestiPradSalygasErlangoMG1();
        void IvestiPradSalygasKoksoGM1();
        void IvestiPradSalygasErlangoGM1();
        float Busenos(int*,int);
        float BusenosKoksoMG1(int *xx, int poz);
        float BusenosErlangoMG1(int *xx, int poz);
        float BusenosKoksoGM1(int *xx, int poz);
        float BusenosErlangoGM1(int *xx, int poz);
        void KasuSk(AnsiString);
        bool PatikrinimasVienatiskumo(int [], int, double);
        void Apjungimas();
        void StacTikRadimas();
        void Statistika_S5();
        void StatistikaKoksoGM1();
        void StatistikaKoksoMG1();
        void Kokso2();
        void Kokso3();
        void Erlango2();
        void Erlango3();
        void Momentai();
        void StringGridas1(int eil, int stulp, double **B);
        void Skaidymas();
        void Kms();
```

```
        void Takahashi();
        void StringGridas2(int eil, int stulp, double *B);
        void StringGridas3(int eil, int stulp, double *B);
        double** Nuline_Matrica(int eil, int stulp);
        double** Vienetine_Matrica(int eil);
        double** Matricu_Daugyba(int eil_a, int stulp_a, double **A, int eil_b,
                          int stulp_b, double **B);
        double** Matricu_Atimtis(int eil, int stulp, double **A, double **B);
        double** Matricu_Sudetis(int eil, int stulp, double **A, double **B);
        double** Transponuota(int eil, int stulp, double **A);
        double** Atvirkstine(int eile, double **A);
        double** Matricos_Dalis(int rusis, int nuo, int iki, double **A);
        double** Gauss_Seidel(int m, double **A, double **X0, double ** B);
public:          // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
//---------------------------------------------------------------------------------------------------------------
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <fstream>
#include <iomanip.h>
using namespace std;

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
 TPr=NULL;
 TBPr=NULL;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button7Click(TObject *Sender)
{
 Close();
}
//---------------------------------------------------------------------------
void TForm1::Normavimas()    //sunormuoja TPr sarasa
{
 TBusIntens *D;
 float suma;    //suma visu iseinanciu is tiriamosios busenos
 int i;
 for (i=0;i<N;i++)
 {
  suma=0;
  D=TPr;
  while (D)
  {
   if (D->BI.eil==i)
    suma=suma+D->BI.BInt;
   D=D->kitas;
  }

  D=TPr;
  while (D)
  {
   if (D->BI.eil==i)
    D->BI.BInt=D->BI.BInt/suma;
   D=D->kitas;
  }
```

```
 }
}
//---------------------------------------------------------------------
void TForm1::Apjungimas()    //sudeda pasikartojancius intensyvumus pereinant is vienos busenos i kita
{
 TBusIntens *D0;
 TBusIntens *D1;
 TBusIntens *D;
 TBusIntens *istr;
 D1=TPr;
 D=TPr;
 int i;
 while (D)
 {
  D0=D;
  D1=D->kitas;
  while (D1)
  {
   if(D->BI.eil==D1->BI.eil)
    if(D->BI.st==D1->BI.st)
     {
      D->BI.BInt=D->BI.BInt+D1->BI.BInt;
      istr=D1;
      D0->kitas=D1->kitas;
      D1=D1->kitas;
      delete istr;
     }
    else {D1=D1->kitas;D0=D0->kitas;}
  else {D1=D1->kitas;D0=D0->kitas;}
  }
 D=D->kitas;
 }
}
//---------------------------------------------------------------------------
bool TForm1::ErgodiskumoPatikrinimas()    //patikrina ar is bet kurios busenos galima patekti i bet
kuria kita
{
 int i=0;
 bool yra=false;
 TBusIntens *X,*Y;
 int *A=new int[N];
 for (i=0;i<N;i++) A[i]=i;      //susirasom visus intervalus
 for (i=0;i<N;i++)
 {
  yra=false;
  X=TPr;
  while ((X)&&(!yra))
  {
   if ((X->BI.eil==A[i])||(X->BI.st==A[i])) yra=true;
   X=X->kitas;
  }
  if (!yra) A[i]=-2;  //tokios busenos/intervalo nera
 }

 yra=false;
 int k=-1;
 while (k!=0)
 {
  if (k==-1) A[N-1]=-1; //pradedam eiti nuo galo
  else A[0]=-1;         //nuo pradziu
  while (!yra)
  {
   yra=true;
   for (i=N-1;i>=0;i--)
   {
    if (A[i]==-1)
    {
     X=TPr;
     while (X)
     {
      if (X->BI.eil==i)
       if (A[X->BI.st]!=-1) {A[X->BI.st]=-1; yra=yra*false;}
      X=X->kitas;
     }
    } //if (A[i]==-1) pab
   } //for (i=N-1;i>=0;i--) pab
  } //while (!yra) pab
  k=0;
 }
 yra=true;     //patikrinam ar i visas busenas sugebejom nueiti
```

```
 i=0;
 while ((i<N)&&(yra))
 {
  if ((A[i]!=-1)&&(A[i]!=-2)) yra=false;
  i++;
 }
if(yra) return true;  //visos busenos yra griztamos
else return false;
}
//---------------------------------------------------------------------
void TForm1::StacTikRadimas()    //kilti lygiu galima tik su liambda wadinas
{                                //vadinasi visada bus tik vienas atejimas ir keli isejimai.gal ce ir
blogai..
 r=new float[N];
 float *S;         //S yra d elementu sarasas
 S=new float[N];
 int i=N-1;
 float a,b,c,d;              //formules nariai
 TBusIntens *P,*X,*ir;
 int b1,b2;               //busena su kuria trinama busena turi santykiu
 float per;                //pasikeites perejimu intensyvumas suskaiciuotas su a,b,c,d
 bool yra;
 int kzk;               //parodo kelinta "iseinama" intensyvuma pasirinkti
 int kzzk;              //neimam reikiamo intensyvumo kol kzzk!=kzk;

//susikuriu nauja sarasa su tais paciais elementais is TPr
 TBusIntens *xx=TPr,*Pp,*Pg;
 Pg=new TBusIntens;        //pirmas elem
 Pg->BI.BInt=xx->BI.BInt;
 Pg->BI.eil=xx->BI.eil;
 Pg->BI.st=xx->BI.st;
 Pg->kitas=NULL;
 P=Pg;
 xx=xx->kitas;

 while (xx)
 {
  Pp=new TBusIntens;
  Pp->BI.BInt=xx->BI.BInt;
  Pp->BI.eil=xx->BI.eil;
  Pp->BI.st=xx->BI.st;
  Pp->kitas=NULL;
  Pg->kitas=Pp;
  Pg=Pg->kitas;
  xx=xx->kitas;
 }
 while (i>0)
 {
  b1=0;
  b2=0;

  d=0;
  X=P;
  kzk=0;
  while (X)    //surandam paskutini formules elementa
  {
    if (X->BI.eil==i)
     if (X->BI.st!=i)
      d=d+X->BI.BInt;
    X=X->kitas;
  }

  while ((b1==0)&&(b2==0))
  {
    b1=-1;
    b2=-1;
    a=0;
    b=0;
    c=0;

  X=P;
  while ((X)&&(b1==-1))    //busena is kurios galim patekti i trinama busena
  {
    if (X->BI.st==i)
     if (X->BI.eil!=i)   //apsauga nuo rinkiu
      {
       b1=X->BI.eil;
       b=X->BI.BInt;
      }
    X=X->kitas;
```

```
 }


 X=P;
 kzzk=0;
 while ((X)&&(b2==-1))   //busena i kuria galim patekti is trinama busena
 {
   if (X->BI.eil==i)
    if (X->BI.st!=i)
    {
     if (kzzk==kzk)
     {
      b2=X->BI.st;
      c=X->BI.BInt;
     }
     kzzk++;
    }
   X=X->kitas;
 }


 X=P;
 while ((X)&&(a==0))    //surandam pirmaji formules elementa
 {
   if ((X->BI.eil==b1)&&(X->BI.st==b2))
   {
    a=X->BI.BInt;
   }
   X=X->kitas;
 }

 if (d!=0) per=a+b*c/d;
 else per=0;

 if ((per>0)&&(b2!=-1))
  {
  //patikrinam ar tokio dar ner
   X=P;
   yra=false;
   while ((X)&&(!yra))
   {
     if (X->BI.eil==b1)
      if (X->BI.st==b2)
      {
       X->BI.BInt=per;
       yra=true;
      }
     X=X->kitas;
   }
 //irasom nauja gauta perejimu intensyvuma
  if (!yra)
  {
   ir=new TBusIntens;
   ir->BI.BInt=per;
   ir->BI.eil=b1;
   ir->BI.st=b2;
   ir->kitas=P;
   P=ir;
  }

//reik paziuret ar tokio dar intensyvumo dar nera pagrindiniam-pradiniam sarase
//ir irasyti jei nera
   X=TPr;
   yra=false;
   while ((X)&&(!yra))
   {
     if (X->BI.eil==b1)
      if (X->BI.st==b2)
      {
       X->BI.BInt=per;
       yra=true;
      }
     X=X->kitas;
   }
 //irasom nauja gauta perejimu intensyvuma
  if (!yra)
  {
   ir=new TBusIntens;
   ir->BI.BInt=per;
   ir->BI.eil=b1;
```

```
  ir->BI.st=b2;
  ir->kitas=TPr;
  TPr=ir;
 }

 b1=0;    //kad dar karta wykdytu cikla
 b2=0;
 kzk++;
}// if per>0 pabaiga

//istrinam ta intensyvuma kuriuo atejom i trinama busena
if ((b2==-1)&&(b1!=-1))
{
 X=P;
 yra=false;
 TBusIntens *Y=NULL,*Z;
 while ((X)&&(!yra))
 {
   if (X->BI.st==i)
    if (X->BI.eil!=i)
    {
     if (Y)
     {
      Y->kitas=X->kitas;
      Z=X;
      delete Z;
      yra=true;
     }
    else   //jeigu reik trinti pask elementa
    {
     Z=X;
     X=X->kitas;
     P=P->kitas;
     delete Z;
     yra=true;
    }
   }
   Y=X;
   if (X->kitas) X=X->kitas;
 }
b1=0;
b2=0;
kzk=0;
} //if b2==-1 pab

//jeigu neradom jokio "ieinancio" inetnsyvumo istrinam visus "iseinancius" inetnsyvumus
if(b1==-1)
{
 X=P;
 TBusIntens *Y=NULL,*Z;
 while (X)
 {
  yra=false;
   if (X->BI.eil==i)
   {
    if (Y)
    {
     Y->kitas=X->kitas;
     Z=X;
     delete Z;
     yra=true;
    }
    else   //jeigu reik trinti pask elementa
    {
     Z=X;
     X=X->kitas;
     P=P->kitas;
     delete Z;
     yra=true;
    }
   }
   if (!yra)
   {
    Y=X;
    X=X->kitas;
   }
   else if (Y) X=Y->kitas;
 }
}     //if b1==-1 pab
```

```
 }      //while (b1==0)&&(b2==0) pabaiga

 S[i]=d;

 i--;
 }

 //r saraso formavimas
 i=1;
 int j;
 float rr;      //busimas r
 r[0]=P->BI.BInt;
 while (i<N)
 {
  rr=0;
  X=TPr;

   while (X)
   {
     if (X->BI.st==i)
      if (X->BI.eil<i)    //nepriklauso rinkes
      {
        rr=rr+r[X->BI.eil]*X->BI.BInt/S[i];
      }
     X=X->kitas;
   }
  r[i]=rr;
  i++;
 }

 float suma=0;
 for(j=0;j<N;j++)
  suma=suma+r[j];
 for(j=0;j<N;j++)
  r[j]=r[j]/suma;

}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
void TForm1::IvestiPradSalygas()
{
 int i=0;
 AnsiString DF;
 OpenDialog1->Filter="Tekstines bylos (*.txt)|*.txt";
 if (OpenDialog1->Execute() && FileExists(OpenDialog1->FileName))
  {
   DF=OpenDialog1->FileName;
   KasuSk(DF);
  }
 else DF="";

 FILE *F;
 char elem;

 TBus *D;
 D=new TBus;
 D->sk=new int[K];
 Tintens=new float[IK];
  if ((F = fopen(DF.c_str(), "r"))==NULL);
  else
   {
    i=0;
     while (elem!='=') fscanf(F, "%c", &elem);
     fscanf(F, "%c", &elem);
    while (elem!=';')
    {
     fscanf(F, "%d%c", &D->sk[i], &elem);
     i++;
    }
     elem='a';
     i=0;
      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);
     while (elem!=';')
     {
      fscanf(F, "%f%c", &Tintens[i], &elem);
      i++;
     }
      while (elem!='=') fscanf(F, "%c", &elem);
```

```
      fscanf(F, "%c", &elem);
     fscanf(F, "%d %c", &R,&elem);      //srautu skaicius

      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);
     fscanf(F, "%d %c", &M,&elem);      //ivykiu skaicius

  fclose(F);
  D->kitas=TBPr;
  TBPr=D;
  N=1;                  //busenu sk

  //if (CheckBox3->Checked==true) SurastiFaziuIntens();
  }

 L=StrToInt(Edit1->Text); //nuskaitom max parasku skaiciu sistemoje
 if (L<0)
 {
  L=0;
  ShowMessage("Ivedet neigiama maksimalu paraisku kieki sistemoje");
 }

}
//---------------------------------------------------------------------------
void TForm1::IvestiPradSalygasKoksoMG1()
{

 int i=0;
 AnsiString DF;
 DF = "Duomenys/KoksoMG1.txt";
 KasuSk(DF);
 FILE *F;
 char elem;

 TBus *D;
 D=new TBus;
 D->sk=new int[K];
 Tintens=new float[IK];
  if ((F = fopen(DF.c_str(), "r"))==NULL);
  else
   {
    i=0;

     while (elem!='=') fscanf(F, "%c", &elem);
     fscanf(F, "%c", &elem);
    while (elem!=';')
    {
     fscanf(F, "%d%c", &D->sk[i], &elem);
     i++;
    }
     elem='a';
     i=0;
      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);



     while (elem!=';')
     {
      fscanf(F, "%f%c", &Tintens[i], &elem);
      i++;
     }

      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);
     fscanf(F, "%d %c", &R,&elem);      //srautu skaicius

      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);
     fscanf(F, "%d %c", &M,&elem);      //ivykiu skaicius

  fclose(F);
  D->kitas=TBPr;
  TBPr=D;
  N=1;                  //busenu sk

  //if (CheckBox3->Checked==true) SurastiFaziuIntens();
  }

 M = 4;
```

```
 R = 1;
 K = 2;
 //IK = 5;
 L=StrToIntDef(Edit1->Text,25); //nuskaitom max parasku skaiciu sistemoje
 if (L<0)
 {
  L=25;
  ShowMessage("Ivedet neigiama maksimalu paraisku kieki sistemoje.Pagal nutylejima programa priskirs
25");
 }
 lamda=StrToFloat(Edit2->Text);
 if (lamda<0)
 {
  lamda=1;
  ShowMessage("Ivedet neigiama maksimalu eksponentinio skirstinio parametra. Pagal nutylejima programa
priskirs 1");
 }

}
//---------------------------------------------------------------------
void TForm1::IvestiPradSalygasKoksoGM1()
{
 int i=0;
 AnsiString DF;
 DF = "Duomenys/KoksoGM1.txt";
 KasuSk(DF);
 FILE *F;
 char elem;

 TBus *D;
 D=new TBus;
 D->sk=new int[K];
 Tintens=new float[IK];
  if ((F = fopen(DF.c_str(), "r"))==NULL);
  else
   {
    i=0;

     while (elem!='=') fscanf(F, "%c", &elem);
     fscanf(F, "%c", &elem);
    while (elem!=';')
    {
     fscanf(F, "%d%c", &D->sk[i], &elem);
     i++;
    }
     elem='a';
     i=0;
      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);



     while (elem!=';')
     {
      fscanf(F, "%f%c", &Tintens[i], &elem);
      i++;
     }

      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);
     fscanf(F, "%d %c", &R,&elem);      //srautu skaicius

      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);
     fscanf(F, "%d %c", &M,&elem);      //ivykiu skaicius

  fclose(F);
  D->kitas=TBPr;
  TBPr=D;
  N=1;                  //busenu sk

  }
 M = 5;
 R = 1;
 K = 3;
 //IK = 5;
 L=StrToIntDef(Edit1->Text,25); //nuskaitom max parasku skaiciu sistemoje
 if (L<0)
 {
  L=25;
```

```
  ShowMessage("Ivedet neigiama maksimalu paraisku kieki sistemoje.Pagal nutylejima programa priskirs
25");
 }
  lamda=StrToFloat(Edit2->Text);
 if (lamda<0)
 {
  lamda=1;
  ShowMessage("Ivedet neigiama maksimalu eksponentinio skirstinio parametra. Pagal nutylejima programa
priskirs 1");
 }
}
//---------------------------------------------------------------------------
void TForm1::IvestiPradSalygasErlangoMG1()
{
 int i=0;
 AnsiString DF;
 DF = "Duomenys/ErlangoMG1.txt";
 KasuSk(DF);
 FILE *F;
 char elem;

 TBus *D;
 D=new TBus;
 D->sk=new int[K];
 Tintens=new float[IK];
  if ((F = fopen(DF.c_str(), "r"))==NULL);
  else
   {
    i=0;

     while (elem!='=') fscanf(F, "%c", &elem);
     fscanf(F, "%c", &elem);
    while (elem!=';')
    {
     fscanf(F, "%d%c", &D->sk[i], &elem);
     i++;
    }
     elem='a';
     i=0;
      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);



     while (elem!=';')
     {
      fscanf(F, "%f%c", &Tintens[i], &elem);
      i++;
     }

      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);
     fscanf(F, "%d %c", &R,&elem);      //srautu skaicius

      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);
     fscanf(F, "%d %c", &M,&elem);      //ivykiu skaicius

  fclose(F);
  D->kitas=TBPr;
  TBPr=D;
  N=1;                  //busenu sk

  }
 M = 2*erl3_n+3;
R = 1;
 K = 2;
 //IK = 5;
 L=StrToIntDef(Edit1->Text,25); //nuskaitom max parasku skaiciu sistemoje
 if (L<0)
 {
  L=25;
  ShowMessage("Ivedet neigiama maksimalu paraisku kieki sistemoje.Pagal nutylejima programa priskirs
25");
 }
  lamda=StrToFloat(Edit2->Text);
 if (lamda<0)
 {
  lamda=1;
```

```
  ShowMessage("Ivedet neigiama maksimalu eksponentinio skirstinio parametra. Pagal nutylejima programa
priskirs 1");
 }
}
//---------------------------------------------------------------------
void TForm1::IvestiPradSalygasErlangoGM1()
{
 int i=0;
 AnsiString DF;
 DF = "Duomenys/ErlangoGM1.txt";
 KasuSk(DF);
 FILE *F;
 char elem;

 TBus *D;
 D=new TBus;
 D->sk=new int[K];
 Tintens=new float[IK];
  if ((F = fopen(DF.c_str(), "r"))==NULL);
  else
   {
    i=0;

     while (elem!='=') fscanf(F, "%c", &elem);
     fscanf(F, "%c", &elem);
    while (elem!=';')
    {
     fscanf(F, "%d%c", &D->sk[i], &elem);
     i++;
    }
     elem='a';
     i=0;
      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);



     while (elem!=';')
     {
      fscanf(F, "%f%c", &Tintens[i], &elem);
      i++;
     }

      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);
     fscanf(F, "%d %c", &R,&elem);      //srautu skaicius

      while (elem!='=') fscanf(F, "%c", &elem);
      fscanf(F, "%c", &elem);
     fscanf(F, "%d %c", &M,&elem);      //ivykiu skaicius

  fclose(F);
  D->kitas=TBPr;
  TBPr=D;
  N=1;                  //busenu sk

  }
 M = 2*erl3_n+4;
 R = 1;
 K = 3;
 //IK = 5;
 L=StrToIntDef(Edit1->Text,25); //nuskaitom max parasku skaiciu sistemoje
 if (L<0)
 {
  L=25;
  ShowMessage("Ivedet neigiama maksimalu paraisku kieki sistemoje.Pagal nutylejima programa priskirs
25");
 }
  lamda=StrToFloat(Edit2->Text);
 if (lamda<0)
 {
  lamda=1;
  ShowMessage("Ivedet neigiama maksimalu eksponentinio skirstinio parametra. Pagal nutylejima programa
priskirs 1");
 }
}
//---------------------------------------------------------------------
void TForm1::IsvestiMemo()
{
 AnsiString x;
```

```
 TBus *D;
 D=TBPr;
 Memo1->Lines->Add("Busenu aibe su");
 Memo1->Lines->Add("stacionariomis tikimybemis >0.00005:");
 for(int k=0; k<N; k++)
 {
  x="[";
  x=x+IntToStr(D->sk[0]);
  for(int j=1; j<K; j++)
  {
   x=x+";"+IntToStr(D->sk[j]);
  }
  x=x+"]";
  if (r[k]>=0.00005)
   Memo1->Lines->Add(IntToStr(k)+") "+x+" - "+FloatToStrF(r[k],ffFixed,6,4));
//  Memo1->Lines->Add(IntToStr(k)+") "+x);
  D=D->kitas;
 }
/*
 Memo1->Lines->Add("");
 Memo1->Lines->Add("Stacionarios tikimybes (>0.005):");
 int k=0;
 while (k<N)
  {
   if (r[k]>=0.00005)
   Memo1->Lines->Add(IntToStr(k)+") "+FloatToStrF(r[k],ffFixed,15,7));
   k++;
  }
}
//---------------------------------------------------------------------
void TForm1::IsvestiStrGrid()
{ P = new double*[N];
  for(int i=0; i<N; i++)
  { P[i] = new double[N];
  }
  for(int i=0; i<N; i++)
  { for(int j=0; j<N; j++)
    { P[i][j] = 0;
    }
  }
 int i,j;
 TBusIntens *D=TPr;
 StringGrid1->ColCount=N+1;
 StringGrid1->RowCount=N+1;
 for(i=0;i<N;i++)
  StringGrid1->Cells[i+1][0]="Bûsena"+IntToStr(i+1);
 for(i=0;i<N;i++)
  StringGrid1->Cells[0][i+1]="Bûsena"+IntToStr(i+1);
 for(i=1;i<=N;i++)
  for(j=1;j<=N;j++)
   StringGrid1->Cells[i][j]=0;
 float x;
 while (D)
  {
   x=StrToFloat(StringGrid1->Cells[D->BI.st+1][D->BI.eil+1]);
   x=x+D->BI.BInt;
   StringGrid1->Cells[D->BI.st+1][D->BI.eil+1]=FloatToStrF(x,ffFixed,3,2);
   P[D->BI.eil][D->BI.st] = x;
   D=D->kitas;
  }

  for(int i=0; i<N; i++)
  { double eil_suma = 0;
    for(int j=0; j<N; j++)
    { eil_suma = eil_suma + P[i][j];
    }
    for(int j=0; j<N; j++)
    { P[i][j] = P[i][j] / eil_suma;
    }
  }
  nn = N;
  StringGridas1(nn, nn, P);
  ofstream fr("Rezultatai/Perejimu_Matrica.txt");
  for (int i = 0; i < nn ; i++)
  { for (int j = 0; j < nn ; j++)
    { fr << /*setw(6) << setprecision(2) <<*/ P[i][j]<<" ";
    }
    fr<< endl;
  }
  fr.close();
```

```
}
//---------------------------------------------------------------------

//---------------------------------------------------------------------

//---------------------------------------------------------------------
void TForm1::BusenuRadimas(int tipas)
{
 int cikl_kint=0;
 bool yra;                  //ziurim ar yra nors wiena laiswa kasa
 int i,j,ii;
 float ints;                //intensyvumas

 int *x;                  //tiriamosios busenos coo
 TBus *D1;                //pagalbine rodykle
 TBus *D2;                //pagalbine rodykle
 TBus *Dck;                //pagalbine rodykle,rodo i ta bsuena kurios numeris cikl_kint
 TBusIntens *D;            //pagalbine rodykle

 x=new int[K];
 D2=TBPr;
 Dck=TBPr;


 while(cikl_kint<N)
  {
   for (j=0;j<K;j++)
    x[j]=Dck->sk[j];
   yra=false;

//tikrinam srautus
 for (ii=0;ii<R;ii++)
  {
   radom=false;

   if (tipas==1) ints=BusenosKoksoMG1(x,ii);     // BUSENOS !!!
   if (tipas==2) ints=BusenosKoksoGM1(x,ii);
   if (tipas==3) ints = BusenosErlangoMG1(x,ii);
   if (tipas==4) ints = BusenosErlangoGM1(x,ii);
            // BUSENOS
   //ints=0.1;
   if (radom)
   {
    if(!PatikrinimasVienatiskumo(x,cikl_kint,ints))
     {
      D1=new TBus;
      D1->sk=new int[K];
      for(j=0;j<K;j++)
       D1->sk[j]=x[j];
      D1->kitas=NULL;
      D2->kitas=D1;
      D2=D2->kitas;

      D=new TBusIntens;
      D->BI.BInt=ints;
      D->BI.eil=cikl_kint;
      D->BI.st=N;
      D->kitas=TPr;
      TPr=D;
      N++;
     }
    for (j=0;j<K;j++)   //atstatom i Dck->sk busena
     x[j]=Dck->sk[j];
   }
  }

  //tikrinam aptarnavimo aparatus
  for (ii=R;ii<M;ii++)
  {
   radom=false;
   if (tipas==1) ints=BusenosKoksoMG1(x,ii);     // BUSENOS !!!
   if (tipas==2) ints=BusenosKoksoGM1(x,ii);
   if (tipas==3) ints = BusenosErlangoMG1(x,ii);
   if (tipas==4) ints = BusenosErlangoGM1(x,ii);
   //ints=0.1;
   if (radom)
   {
    if(!PatikrinimasVienatiskumo(x,cikl_kint,ints))
     {
```

```
      D1=new TBus;
      D1->sk=new int[K];
      for(j=0;j<K;j++)
       D1->sk[j]=x[j];
      D1->kitas=NULL;
      D2->kitas=D1;
      D2=D2->kitas;

      D=new TBusIntens;
      D->BI.BInt=ints;
      D->BI.eil=cikl_kint;
      D->BI.st=N;
      D->kitas=TPr;
      TPr=D;
      N++;
     }
    for (j=0;j<K;j++)   //atstatom i Dck->sk busena
     x[j]=Dck->sk[j];
   }
  }
  cikl_kint++;
  Dck=Dck->kitas;
  } //while(cikl_kint<N) pab


}

//---------------------------------------------------------------------

float TForm1::Busenos(int *xx, int poz)
{
 switch(poz)
 {
  case 0:
     {
      if( xx[0]==0)
        { xx[0]=1;
          radom = true;
        }
      return 0.5*1000000;
     }  // case 0 pabaiga

  case 1:
     {
      if( xx[0]==0)
        { xx[0]=2;
          radom = true;
        }
      return 0.5*1000000;
     }  // case 0 pabaiga

  case 2:
     {
      if( xx[0]==1)
        {
          if((xx[2]==0)&&(xx[1]<L))
            { xx[0]=0;
              xx[1]++;
              xx[2]=1;
              radom=true;
            }
          else if (xx[1]<L)
            { xx[0]=0;
              xx[1]++;
              radom=true;
            }
        }
      return 1;
     }    //  case 2 pabaiga

    case 3:
     {
      if( xx[0]==2)
        {
          if((xx[2]==0)&&(xx[1]<L))
            { xx[0]=0;
              xx[1]++;
              xx[2]=1;
              radom=true;
            }
```

```
          else  if (xx[1]<L)
            { xx[0]=0;
              xx[1]++;
              radom=true;
            }
        }
      return 0.5;
     }    //  case 2 pabaiga

  case 4:
     {
      if( xx[2]==1)
        {
          if( xx[1]>1)
            { xx[1]--;
              xx[2]=1;
              radom=true;
            }
          else
            { xx[1]--;
              xx[2]=0;
              radom=true;
            }
        }
      return 1;
     }    //  case 2 pabaiga
 }   // switch pabaiga




}  // Busenos pabaiga

//---------------------------------------------------------------------
bool TForm1::PatikrinimasVienatiskumo(int s[], int Bnr, double intens) //Bnr busenos numeris is kurios
atejom i surasta busena
{
 bool v=false;
 TBusIntens *D;
 ///TBusA *D1;
 ///D1=TBAPr;
 TBus *D1;
 D1=TBPr;
 int i=0,j;
 while((i<N)&&(!v))
 {
     j=0;
     v=true;
     while((j<K)&&(v))
      {
       ///if (int(s[j]*100000)!=int(D1->sk[j]*100000))
       if (s[j]!=D1->sk[j])
        {
          v=false;
        }
       j++;
      }
    //}
  D1=D1->kitas;
  i++;
 }

 if (v)
  {
   D=new TBusIntens;
   D->BI.BInt=intens;
   D->BI.eil=Bnr;
   D->BI.st=i-1;
   D->kitas=TPr;
   TPr=D;
  }
 return v;   //jeigu tokia busena jau buwo grazins true
}
//---------------------------------------------------------------------------
float TForm1::BusenosKoksoMG1(int *xx, int poz)
{
    switch(poz)
  {
  case 0:
```

```
    {
     if( (xx[1]==0)&&(xx[0]<L))
       {
         xx[0]++;
         xx[1]=1;
         radom=true;
       }
         else if (xx[0]<L)
           { xx[0]++;
             radom=true;
           }
     return lamda;
    }  // case 0 pabaiga

  case 1:
    {
     if( xx[1]==1)
       { xx[1]=2;
         radom = true;
       }
    return kok3_p*kok3_miu1;
     //return 0.005592;
     //return 0.7075*1.4587;
    }  // case 1 pabaiga

  case 2:
    {
     if( xx[1]==1)
       {
         if( xx[0]>1)
           { xx[0]--;
             xx[1]=1;
             radom=true;
           }
         else
           { xx[0]--;
             xx[1]=0;
             radom=true;
           }
       }
     return (1-kok3_p)*kok3_miu1;
      //return  (1-0.005592)*1.132817;
     //return (1-0.7075)*1.4587;
    }  // case 2 pabaiga

  case 3:
    {
     if( xx[1]==2)
       {
         if( xx[0]>1)
           { xx[0]--;
             xx[1]=1;
             radom=true;
           }
         else
           { xx[0]--;
             xx[1]=0;
             radom=true;
           }
       }
     return kok3_miu2;
     //return 0.210069;
     //return 2.808;
    }    //  case 3 pabaiga

 }   // switch pabaiga


} // BusenosKoksoMG1  pabaiga



//---------------------------------------------------------------------------
float TForm1::BusenosKoksoGM1(int *xx, int poz)
{
    switch(poz)
  {
  case 0:
    {
     if( xx[0]==0)
```

```
       { xx[0]=1;
         radom = true;
       }
       return 100000;
    }  // case 0 pabaiga

  case 1:
     {
      if( xx[0]==1)
       { xx[0]=2;
         radom = true;
       }
       return kok3_p*kok3_miu1;
     //return 0.7075*1.4587;
    }  // case 1 pabaiga

  case 2:
     {
      if( xx[0]==1)
       {
         if((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0;
             xx[1]++;
             xx[2]=1;
             radom=true;
           }
         else if (xx[1]<L)
           { xx[0]=0;
             xx[1]++;
             radom=true;
           }
       }
       return (1-kok3_p)*kok3_miu1;
     //return (1-0.7075)*1.4587;
    }  // case 2 pabaiga


  case 3:
     {
      if( xx[0]==2)
       {
         if((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0;
             xx[1]++;
             xx[2]=1;
             radom=true;
           }
         else if (xx[1]<L)
           { xx[0]=0;
             xx[1]++;
             radom=true;
           }
       }
      return kok3_miu2;
      //return 2.808;
     }    //  case 3 pabaiga

  case 4:
     {
      if( xx[2]==1)
       {
         if( xx[1]>1)
           { xx[1]--;
             xx[2]=1;
             radom=true;
           }
         else
           { xx[1]--;
             xx[2]=0;
             radom=true;
           }
       }
      return lamda;
     }    //  case 4 pabaiga

 }   // switch pabaiga


} // BusenosKoksoMG1  pabaiga
//---------------------------------------------------------------------
```

```
float TForm1::BusenosErlangoGM1(int *xx, int poz)
{
// ----------erlango GM  1 ----------------------------------------
if (erl3_n==1)
{ switch(poz) {
   case 0:
   { if ( xx[0]==0) { xx[0]=1;radom = true; } return 100000; } // case 0 pabaiga
   case 1:
   { if ( xx[0]==1) { xx[0]=2;radom = true;}return erl3_p*100000;} // case 1 pabaiga
   case 2:
   {if ( xx[0]==1) { xx[0]=3;radom = true;} return (1-erl3_p)*100000;} // case 2 pabaiga
   case 3:
   { if ( xx[0]==2) {
        if ((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0; xx[1]++;xx[2]=1;radom = true;}
        else  if (xx[1]<L)
           { xx[0]=0; xx[1]++;radom =true; } } return erl3_miu1; } // case 3 pabaiga
   case 4:
   { if ( xx[0]==3) {
        if ((xx[2]==0)&&(xx[1]<L)){ xx[0]=0; xx[1]++; xx[2]=1;  radom = true; }
        else  if (xx[1]<L) { xx[0]=0;xx[1]++;radom =true;}}return erl3_miu2; } // case 4 pabaiga
   case 5:
   {if ( xx[2]==1) {
        if ( xx[1]>1) { xx[1]--; xx[2]=1; radom = true; }
        else{ xx[1]--;xx[2]=0;radom = true; }} return lamda;}  // case 10 pabaiga
 }  //  switch pabaiga
} // erlano GM 1 pabaiga ----------------------------------------

// ----------erlango GM  2 ----------------------------------------
if (erl3_n==2)
{ switch(poz) {
   case 0:
   { if ( xx[0]==0) { xx[0]=1;radom = true; } return 100000; } // case 0 pabaiga
   case 1:
   { if ( xx[0]==1) { xx[0]=2;radom = true;}return erl3_p*100000;} // case 1 pabaiga
   case 2:
   {if ( xx[0]==1) { xx[0]=3;radom = true;} return (1-erl3_p)*100000;} // case 2 pabaiga
   case 3:
    { if (xx[0]==2) {xx[0]=4;radom=true;} return erl3_miu1;}
   case 4:
    { if (xx[0]==3) {xx[0]=5;radom=true;} return erl3_miu2;}
   case 5:
   { if ( xx[0]==4) {
        if ((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0; xx[1]++;xx[2]=1;radom = true;}
        else  if (xx[1]<L)
           { xx[0]=0; xx[1]++;radom =true; } } return erl3_miu1; } // case 3 pabaiga
   case 6:
   { if ( xx[0]==5) {
        if ((xx[2]==0)&&(xx[1]<L)){ xx[0]=0; xx[1]++; xx[2]=1;  radom = true; }
        else  if (xx[1]<L) { xx[0]=0;xx[1]++;radom =true;}}return erl3_miu2; } // case 4 pabaiga
   case 7:
   {if ( xx[2]==1) {
        if ( xx[1]>1) { xx[1]--; xx[2]=1; radom = true; }
        else{ xx[1]--;xx[2]=0;radom = true; }} return lamda;}  // case 10 pabaiga
 }  //  switch pabaiga
} // erlano GM 2 pabaiga ----------------------------------------------------

// ----------erlango GM  3 ----------------------------------------
if (erl3_n==3)
{ switch(poz) {
   case 0:
   { if ( xx[0]==0) { xx[0]=1;radom = true; } return 100000; } // case 0 pabaiga
   case 1:
   { if ( xx[0]==1) { xx[0]=2;radom = true;}return erl3_p*100000;} // case 1 pabaiga
   case 2:
   {if ( xx[0]==1) { xx[0]=3;radom = true;} return (1-erl3_p)*100000;} // case 2 pabaiga
   case 3:
    { if (xx[0]==2) {xx[0]=4;radom=true;} return erl3_miu1;}
   case 4:
    { if (xx[0]==3) {xx[0]=5;radom=true;} return erl3_miu2;}
    case 5:
    { if (xx[0]==4) {xx[0]=6;radom=true;} return erl3_miu1;}
   case 6:
    { if (xx[0]==5) {xx[0]=7;radom=true;} return erl3_miu2;}
   case 7:
   { if ( xx[0]==6) {
        if ((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0; xx[1]++;xx[2]=1;radom = true;}
        else  if (xx[1]<L)
```

```
           { xx[0]=0; xx[1]++;radom =true; } } return erl3_miu1; } // case 3 pabaiga
   case 8:
   { if ( xx[0]==7) {
        if ((xx[2]==0)&&(xx[1]<L)){ xx[0]=0; xx[1]++; xx[2]=1;  radom = true; }
        else  if (xx[1]<L) { xx[0]=0;xx[1]++;radom =true;}}return erl3_miu2; } // case 4 pabaiga
   case 9:
   {if ( xx[2]==1) {
        if ( xx[1]>1) { xx[1]--; xx[2]=1; radom = true; }
        else{ xx[1]--;xx[2]=0;radom = true; }} return lamda;}  // case 10 pabaiga
 }  //  switch pabaiga
} // erlano GM 3 pabaiga ----------------------------------------------------

// ----------erlango GM  4 ----------------------------------------
if (erl3_n==4)
{ switch(poz) {
   case 0:
   { if ( xx[0]==0) { xx[0]=1;radom = true; } return 100000; } // case 0 pabaiga
   case 1:
   { if ( xx[0]==1) { xx[0]=2;radom = true;}return erl3_p*100000;} // case 1 pabaiga
   case 2:
   {if ( xx[0]==1) { xx[0]=3;radom = true;} return (1-erl3_p)*100000;} // case 2 pabaiga
   case 3:
    { if (xx[0]==2) {xx[0]=4;radom=true;} return erl3_miu1;}
   case 4:
    { if (xx[0]==3) {xx[0]=5;radom=true;} return erl3_miu2;}
    case 5:
    { if (xx[0]==4) {xx[0]=6;radom=true;} return erl3_miu1;}
   case 6:
    { if (xx[0]==5) {xx[0]=7;radom=true;} return erl3_miu2;}
     case 7:
    { if (xx[0]==6) {xx[0]=8;radom=true;} return erl3_miu1;}
   case 8:
    { if (xx[0]==7) {xx[0]=9;radom=true;} return erl3_miu2;}
   case 9:
   { if ( xx[0]==8) {
        if ((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0; xx[1]++;xx[2]=1;radom = true;}
        else  if (xx[1]<L)
           { xx[0]=0; xx[1]++;radom =true; } } return erl3_miu1; } // case 3 pabaiga
   case 10:
   { if ( xx[0]==9) {
        if ((xx[2]==0)&&(xx[1]<L)){ xx[0]=0; xx[1]++; xx[2]=1;  radom = true; }
        else  if (xx[1]<L) { xx[0]=0;xx[1]++;radom =true;}}return erl3_miu2; } // case 4 pabaiga
   case 11:
   {if ( xx[2]==1) {
        if ( xx[1]>1) { xx[1]--; xx[2]=1; radom = true; }
        else{ xx[1]--;xx[2]=0;radom = true; }} return lamda;}  // case 10 pabaiga
 }  //  switch pabaiga
} // erlano GM 4 pabaiga ----------------------------------------------------

// ----------erlango GM  5 ----------------------------------------
if (erl3_n==5)
{ switch(poz) {
   case 0:
   { if ( xx[0]==0) { xx[0]=1;radom = true; } return 100000; } // case 0 pabaiga
   case 1:
   { if ( xx[0]==1) { xx[0]=2;radom = true;}return erl3_p*100000;} // case 1 pabaiga
   case 2:
   {if ( xx[0]==1) { xx[0]=3;radom = true;} return (1-erl3_p)*100000;} // case 2 pabaiga
   case 3:
    { if (xx[0]==2) {xx[0]=4;radom=true;} return erl3_miu1;}
   case 4:
    { if (xx[0]==3) {xx[0]=5;radom=true;} return erl3_miu2;}
   case 5:
    { if (xx[0]==4) {xx[0]=6;radom=true;} return erl3_miu1;}
   case 6:
    { if (xx[0]==5) {xx[0]=7;radom=true;} return erl3_miu2;}
   case 7:
    { if (xx[0]==6) {xx[0]=8;radom=true;} return erl3_miu1;}
   case 8:
    { if (xx[0]==7) {xx[0]=9;radom=true;} return erl3_miu2;}
   case 9:
    { if (xx[0]==8) {xx[0]=10;radom=true;} return erl3_miu1;}
   case 10:
    { if (xx[0]==9) {xx[0]=11;radom=true;} return erl3_miu2;}
   case 11:
   { if ( xx[0]==10) {
        if ((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0; xx[1]++;xx[2]=1;radom = true;}
        else  if (xx[1]<L)
```

```
          { xx[0]=0; xx[1]++;radom =true; } } return erl3_miu1; } // case 3 pabaiga
   case 12:
   { if ( xx[0]==11) {
        if ((xx[2]==0)&&(xx[1]<L)){ xx[0]=0; xx[1]++; xx[2]=1;  radom = true; }
        else  if (xx[1]<L) { xx[0]=0;xx[1]++;radom =true;}}return erl3_miu2; } // case 4 pabaiga
   case 13:
   {if ( xx[2]==1) {
        if ( xx[1]>1) { xx[1]--; xx[2]=1; radom = true; }
        else{ xx[1]--;xx[2]=0;radom = true; }} return lamda;}  // case 10 pabaiga
 }  //  switch pabaiga
} // erlano GM 5 pabaiga ------------------------------------------------------

// ----------erlango GM  6 ----------------------------------------
if (erl3_n==6)
{ switch(poz) {
   case 0:
   { if ( xx[0]==0) { xx[0]=1;radom = true; } return 100000; } // case 0 pabaiga
   case 1:
   { if ( xx[0]==1) { xx[0]=2;radom = true;}return erl3_p*100000;} // case 1 pabaiga
   case 2:
   {if ( xx[0]==1) { xx[0]=3;radom = true;} return (1-erl3_p)*100000;} // case 2 pabaiga
   case 3:
    { if (xx[0]==2) {xx[0]=4;radom=true;} return erl3_miu1;}
   case 4:
    { if (xx[0]==3) {xx[0]=5;radom=true;} return erl3_miu2;}
   case 5:
    { if (xx[0]==4) {xx[0]=6;radom=true;} return erl3_miu1;}
   case 6:
    { if (xx[0]==5) {xx[0]=7;radom=true;} return erl3_miu2;}
   case 7:
    { if (xx[0]==6) {xx[0]=8;radom=true;} return erl3_miu1;}
   case 8:
    { if (xx[0]==7) {xx[0]=9;radom=true;} return erl3_miu2;}
   case 9:
    { if (xx[0]==8) {xx[0]=10;radom=true;} return erl3_miu1;}
   case 10:
    { if (xx[0]==9) {xx[0]=11;radom=true;} return erl3_miu2;}
   case 11:
    { if (xx[0]==10) {xx[0]=12;radom=true;} return erl3_miu1;}
   case 12:
    { if (xx[0]==11) {xx[0]=13;radom=true;} return erl3_miu2;}
   case 13:
   { if ( xx[0]==12) {
        if ((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0; xx[1]++;xx[2]=1;radom = true;}
        else  if (xx[1]<L)
           { xx[0]=0; xx[1]++;radom =true; } } return erl3_miu1; } // case 3 pabaiga
   case 14:
   { if ( xx[0]==13) {
        if ((xx[2]==0)&&(xx[1]<L)){ xx[0]=0; xx[1]++; xx[2]=1;  radom = true; }
        else  if (xx[1]<L) { xx[0]=0;xx[1]++;radom =true;}}return erl3_miu2; } // case 4 pabaiga
   case 15:
   {if ( xx[2]==1) {
        if ( xx[1]>1) { xx[1]--; xx[2]=1; radom = true; }
        else{ xx[1]--;xx[2]=0;radom = true; }} return lamda;}  // case 10 pabaiga
 }  //  switch pabaiga
} // erlano GM 6 pabaiga ------------------------------------------------------

// ----------erlango GM  7 ----------------------------------------
if (erl3_n==7)
{ switch(poz) {
   case 0:
   { if ( xx[0]==0) { xx[0]=1;radom = true; } return 100000; } // case 0 pabaiga
   case 1:
   { if ( xx[0]==1) { xx[0]=2;radom = true;}return erl3_p*100000;} // case 1 pabaiga
   case 2:
   {if ( xx[0]==1) { xx[0]=3;radom = true;} return (1-erl3_p)*100000;} // case 2 pabaiga
   case 3:
    { if (xx[0]==2) {xx[0]=4;radom=true;} return erl3_miu1;}
   case 4:
    { if (xx[0]==3) {xx[0]=5;radom=true;} return erl3_miu2;}
   case 5:
    { if (xx[0]==4) {xx[0]=6;radom=true;} return erl3_miu1;}
   case 6:
    { if (xx[0]==5) {xx[0]=7;radom=true;} return erl3_miu2;}
   case 7:
    { if (xx[0]==6) {xx[0]=8;radom=true;} return erl3_miu1;}
   case 8:
    { if (xx[0]==7) {xx[0]=9;radom=true;} return erl3_miu2;}
   case 9:
```

```
    { if (xx[0]==8) {xx[0]=10;radom=true;} return erl3_miu1;}
   case 10:
    { if (xx[0]==9) {xx[0]=11;radom=true;} return erl3_miu2;}
   case 11:
    { if (xx[0]==10) {xx[0]=12;radom=true;} return erl3_miu1;}
   case 12:
    { if (xx[0]==11) {xx[0]=13;radom=true;} return erl3_miu2;}
   case 13:
    { if (xx[0]==12) {xx[0]=14;radom=true;} return erl3_miu1;}
   case 14:
    { if (xx[0]==13) {xx[0]=15;radom=true;} return erl3_miu2;}
   case 15:
   { if ( xx[0]==14) {
        if ((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0; xx[1]++;xx[2]=1;radom = true;}
        else  if (xx[1]<L)
           { xx[0]=0; xx[1]++;radom =true; } } return erl3_miu1; } // case 3 pabaiga
   case 16:
   { if ( xx[0]==15) {
        if ((xx[2]==0)&&(xx[1]<L)){ xx[0]=0; xx[1]++; xx[2]=1;  radom = true; }
        else  if (xx[1]<L) { xx[0]=0;xx[1]++;radom =true;}}return erl3_miu2; } // case 4 pabaiga
   case 17:
   {if ( xx[2]==1) {
        if ( xx[1]>1) { xx[1]--; xx[2]=1; radom = true; }
        else{ xx[1]--;xx[2]=0;radom = true; }} return lamda;}  // case 10 pabaiga
 }  //  switch pabaiga
} // erlano GM 7 pabaiga ----------------------------------------------------

// ----------erlango GM  8------------------------------------------------------
if (erl3_n==8)
{ switch(poz) {
   case 0:
   { if ( xx[0]==0) { xx[0]=1;radom = true; } return 100000; } // case 0 pabaiga
   case 1:
   { if ( xx[0]==1) { xx[0]=2;radom = true;}return erl3_p*100000;} // case 1 pabaiga
   case 2:
   {if ( xx[0]==1) { xx[0]=3;radom = true;} return (1-erl3_p)*100000;} // case 2 pabaiga
   case 3:
    { if (xx[0]==2) {xx[0]=4;radom=true;} return erl3_miu1;}
   case 4:
    { if (xx[0]==3) {xx[0]=5;radom=true;} return erl3_miu2;}
   case 5:
    { if (xx[0]==4) {xx[0]=6;radom=true;} return erl3_miu1;}
   case 6:
    { if (xx[0]==5) {xx[0]=7;radom=true;} return erl3_miu2;}
   case 7:
    { if (xx[0]==6) {xx[0]=8;radom=true;} return erl3_miu1;}
   case 8:
    { if (xx[0]==7) {xx[0]=9;radom=true;} return erl3_miu2;}
   case 9:
    { if (xx[0]==8) {xx[0]=10;radom=true;} return erl3_miu1;}
   case 10:
    { if (xx[0]==9) {xx[0]=11;radom=true;} return erl3_miu2;}
   case 11:
    { if (xx[0]==10) {xx[0]=12;radom=true;} return erl3_miu1;}
   case 12:
    { if (xx[0]==11) {xx[0]=13;radom=true;} return erl3_miu2;}
   case 13:
    { if (xx[0]==12) {xx[0]=14;radom=true;} return erl3_miu1;}
   case 14:
    { if (xx[0]==13) {xx[0]=15;radom=true;} return erl3_miu2;}
   case 15:
    { if (xx[0]==14) {xx[0]=16;radom=true;} return erl3_miu1;}
   case 16:
    { if (xx[0]==15) {xx[0]=17;radom=true;} return erl3_miu2;}
   case 17:
   { if ( xx[0]==16) {
        if ((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0; xx[1]++;xx[2]=1;radom = true;}
        else  if (xx[1]<L)
           { xx[0]=0; xx[1]++;radom =true; } } return erl3_miu1; } // case 3 pabaiga
   case 18:
   { if ( xx[0]==17) {
        if ((xx[2]==0)&&(xx[1]<L)){ xx[0]=0; xx[1]++; xx[2]=1;  radom = true; }
        else  if (xx[1]<L) { xx[0]=0;xx[1]++;radom =true;}}return erl3_miu2; } // case 4 pabaiga
   case 19:
   {if ( xx[2]==1) {
        if ( xx[1]>1) { xx[1]--; xx[2]=1; radom = true; }
        else{ xx[1]--;xx[2]=0;radom = true; }} return lamda;}  // case 10 pabaiga
 }  //  switch pabaiga
```

```
} // erlango GM 8 pabaiga ----------------------------------------------------

// ----------erlango GM  9----------------------------------------------------
if (erl3_n==9)
{ switch(poz) {
   case 0:
   { if ( xx[0]==0) { xx[0]=1;radom = true; } return 100000; } // case 0 pabaiga
   case 1:
   { if ( xx[0]==1) { xx[0]=2;radom = true;}return erl3_p*100000;} // case 1 pabaiga
   case 2:
   {if ( xx[0]==1) { xx[0]=3;radom = true;} return (1-erl3_p)*100000;} // case 2 pabaiga
   case 3:
    { if (xx[0]==2) {xx[0]=4;radom=true;} return erl3_miu1;}
   case 4:
    { if (xx[0]==3) {xx[0]=5;radom=true;} return erl3_miu2;}
   case 5:
    { if (xx[0]==4) {xx[0]=6;radom=true;} return erl3_miu1;}
   case 6:
    { if (xx[0]==5) {xx[0]=7;radom=true;} return erl3_miu2;}
   case 7:
    { if (xx[0]==6) {xx[0]=8;radom=true;} return erl3_miu1;}
   case 8:
    { if (xx[0]==7) {xx[0]=9;radom=true;} return erl3_miu2;}
   case 9:
    { if (xx[0]==8) {xx[0]=10;radom=true;} return erl3_miu1;}
   case 10:
    { if (xx[0]==9) {xx[0]=11;radom=true;} return erl3_miu2;}
   case 11:
    { if (xx[0]==10) {xx[0]=12;radom=true;} return erl3_miu1;}
   case 12:
    { if (xx[0]==11) {xx[0]=13;radom=true;} return erl3_miu2;}
   case 13:
    { if (xx[0]==12) {xx[0]=14;radom=true;} return erl3_miu1;}
   case 14:
    { if (xx[0]==13) {xx[0]=15;radom=true;} return erl3_miu2;}
   case 15:
    { if (xx[0]==14) {xx[0]=16;radom=true;} return erl3_miu1;}
   case 16:
    { if (xx[0]==15) {xx[0]=17;radom=true;} return erl3_miu2;}
   case 17:
    { if (xx[0]==16) {xx[0]=18;radom=true;} return erl3_miu1;}
   case 18:
    { if (xx[0]==17) {xx[0]=19;radom=true;} return erl3_miu2;}
   case 19:
   { if ( xx[0]==18) {
        if ((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0; xx[1]++;xx[2]=1;radom = true;}
        else  if (xx[1]<L)
           { xx[0]=0; xx[1]++;radom =true; } } return erl3_miu1; } // case 3 pabaiga
   case 20:
   { if ( xx[0]==19) {
        if ((xx[2]==0)&&(xx[1]<L)){ xx[0]=0; xx[1]++; xx[2]=1;  radom = true; }
        else  if (xx[1]<L) { xx[0]=0;xx[1]++;radom =true;}}return erl3_miu2; } // case 4 pabaiga
   case 21:
   {if ( xx[2]==1) {
        if ( xx[1]>1) { xx[1]--; xx[2]=1; radom = true; }
        else{ xx[1]--;xx[2]=0;radom = true; }} return lamda;}  // case 10 pabaiga
 }  //  switch pabaiga
} // erlano GM 9 pabaiga ----------------------------------------------------

// ----------erlango GM  10---------------------------------------------------
if (erl3_n==10)
{ switch(poz) {
   case 0:
   { if ( xx[0]==0) { xx[0]=1;radom = true; } return 100000; } // case 0 pabaiga
   case 1:
   { if ( xx[0]==1) { xx[0]=2;radom = true;}return erl3_p*100000;} // case 1 pabaiga
   case 2:
   {if ( xx[0]==1) { xx[0]=3;radom = true;} return (1-erl3_p)*100000;} // case 2 pabaiga
   case 3:
    { if (xx[0]==2) {xx[0]=4;radom=true;} return erl3_miu1;}
   case 4:
    { if (xx[0]==3) {xx[0]=5;radom=true;} return erl3_miu2;}
   case 5:
    { if (xx[0]==4) {xx[0]=6;radom=true;} return erl3_miu1;}
   case 6:
    { if (xx[0]==5) {xx[0]=7;radom=true;} return erl3_miu2;}
   case 7:
    { if (xx[0]==6) {xx[0]=8;radom=true;} return erl3_miu1;}
   case 8:
```

```
    { if (xx[0]==7) {xx[0]=9;radom=true;} return erl3_miu2;}
   case 9:
    { if (xx[0]==8) {xx[0]=10;radom=true;} return erl3_miu1;}
   case 10:
    { if (xx[0]==9) {xx[0]=11;radom=true;} return erl3_miu2;}
   case 11:
    { if (xx[0]==10) {xx[0]=12;radom=true;} return erl3_miu1;}
   case 12:
    { if (xx[0]==11) {xx[0]=13;radom=true;} return erl3_miu2;}
   case 13:
    { if (xx[0]==12) {xx[0]=14;radom=true;} return erl3_miu1;}
   case 14:
    { if (xx[0]==13) {xx[0]=15;radom=true;} return erl3_miu2;}
   case 15:
    { if (xx[0]==14) {xx[0]=16;radom=true;} return erl3_miu1;}
   case 16:
    { if (xx[0]==15) {xx[0]=17;radom=true;} return erl3_miu2;}
   case 17:
    { if (xx[0]==16) {xx[0]=18;radom=true;} return erl3_miu1;}
   case 18:
    { if (xx[0]==17) {xx[0]=19;radom=true;} return erl3_miu2;}
   case 19:
    { if (xx[0]==18) {xx[0]=20;radom=true;} return erl3_miu1;}
   case 20:
    { if (xx[0]==19) {xx[0]=21;radom=true;} return erl3_miu2;}
   case 21:
   { if ( xx[0]==20) {
        if ((xx[2]==0)&&(xx[1]<L))
           { xx[0]=0; xx[1]++;xx[2]=1;radom = true;}
        else  if (xx[1]<L)
           { xx[0]=0; xx[1]++;radom =true; } } return erl3_miu1; } // case 3 pabaiga
   case 22:
   { if ( xx[0]==21) {
        if ((xx[2]==0)&&(xx[1]<L)){ xx[0]=0; xx[1]++; xx[2]=1;  radom = true; }
        else  if (xx[1]<L) { xx[0]=0;xx[1]++;radom =true;}}return erl3_miu2; } // case 4 pabaiga
   case 23:
   {if ( xx[2]==1) {
        if ( xx[1]>1) { xx[1]--; xx[2]=1; radom = true; }
        else{ xx[1]--;xx[2]=0;radom = true; }} return lamda;}  // case 10 pabaiga
 }  //  switch pabaiga
} // erlano GM 10 pabaiga ----------------------------------------------------



} // BusenosErlangoGM1  pabaiga

//---------------------------------------------------------------------
float TForm1::BusenosErlangoMG1(int *xx, int poz)
{
 // erlango gm1 1 ------------------------------------------------------------
 if (erl3_n==1)
 {
   switch(poz)
   {
    case 0:
    {
     if( (xx[1]==0)&&(xx[0]<L))
        {
          xx[0]++;
          xx[1]=1;
          radom=true;
        }
          else if (xx[0]<L)
            { xx[0]++;
              radom=true;
            }
      return lamda;
    } // case 0 pabaiga

    case 1:
    {
      if (xx[1]==1)
         {
          xx[1]=2;
          radom = true;
         }
       return erl3_p*100000;
    } // case 1 pabaiga

    case 2:
```

```
   {
     if (xx[1]==1)
        {
         xx[1]=3;
         radom = true;
        }
      return (1-erl3_p)*100000;
   } // case 2 pabaiga

   case 3:
   {
    if ( xx[1]==2)
      {
       if ( xx[0]>1)
          { xx[0]--;
            xx[1]=1;
            radom = true;
          }
       else
          { xx[0]--;
            xx[1]=0;
            radom = true;
          }
      }
    return erl3_miu1;
   } // case 3 pabaiga

   case 4:
   {
    if ( xx[1]==3)
      {
       if ( xx[0]>1)
          { xx[0]--;
            xx[1]=1;
            radom = true;
          }
       else
          { xx[0]--;
            xx[1]=0;
            radom = true;
          }
      }
    return erl3_miu2;
    Memo1->Lines->Add("Yoo");
   } // case 4 pabaiga

  }     // switch pabaiga
}  // erl3_n lygu 1


if (erl3_n==2)
{
 switch(poz)
  {
   case 0:
   {
    if( (xx[1]==0)&&(xx[0]<L))
       {
         xx[0]++;
         xx[1]=1;
         radom=true;
       }
         else if (xx[0]<L)
           { xx[0]++;
             radom=true;
           }
     return lamda;
   } // case 0 pabaiga

   case 1:
   {
     if (xx[1]==1)
        {
         xx[1]=2;
         radom = true;
        }
      return erl3_p*100000;
   } // case 1 pabaiga

   case 2:
```

```
    {
      if (xx[1]==1)
          {
           xx[1]=3;
           radom = true;
          }
       return (1-erl3_p)*100000;
    } // case 2 pabaiga

     case 3:
    {
      if (xx[1]==2)
          {
           xx[1]=4;
           radom = true;
          }
       return erl3_miu1;
    } // case 3 pabaiga

     case 4:
    {
      if (xx[1]==3)
          {
           xx[1]=5;
           radom = true;
          }
       return erl3_miu2;
    } // case 4 pabaiga

    case 5:
    {
     if ( xx[1]==4)
        {
         if ( xx[0]>1)
            { xx[0]--;
              xx[1]=1;
              radom = true;
            }
         else
            { xx[0]--;
              xx[1]=0;
              radom = true;
            }
        }
     return erl3_miu1;
    } // case 5 pabaiga

    case 6:
    {
     if ( xx[1]==5)
        {
         if ( xx[0]>1)
            { xx[0]--;
              xx[1]=1;
              radom = true;
            }
         else
            { xx[0]--;
              xx[1]=0;
              radom = true;
            }
        }
     return erl3_miu2;
    } // case 6 pabaiga

   }     // switch pabaiga
 }  // erl3_n lygu 2

// ----------------------erlango 3-----------------------------------------
 if (erl3_n==3)
 {
  switch(poz)
   {
    case 0:
    {
     if( (xx[1]==0)&&(xx[0]<L))
        {
          xx[0]++;
          xx[1]=1;
          radom=true;
```

```
    }
      else if (xx[0]<L)
        { xx[0]++;
          radom=true;
        }
  return lamda;
} // case 0 pabaiga

case 1:
{
  if (xx[1]==1)
     {
      xx[1]=2;
      radom = true;
     }
   return erl3_p*100000;
} // case 1 pabaiga

case 2:
{
  if (xx[1]==1)
     {
      xx[1]=3;
      radom = true;
     }
   return (1-erl3_p)*100000;
} // case 2 pabaiga

 case 3:
{
  if (xx[1]==2)
     {
      xx[1]=4;
      radom = true;
     }
   return erl3_miu1;
} // case 3 pabaiga

 case 4:
{
  if (xx[1]==3)
     {
      xx[1]=5;
      radom = true;
     }
   return erl3_miu2;
} // case 4 pabaiga

 case 5:
{
  if (xx[1]==4)
     {
      xx[1]=6;
      radom = true;
     }
   return erl3_miu1;
} // case 5 pabaiga

 case 6:
{
  if (xx[1]==5)
     {
      xx[1]=7;
      radom = true;
     }
   return erl3_miu2;
} // case 6 pabaiga


case 7:
{
 if ( xx[1]==6)
   {
    if ( xx[0]>1)
       { xx[0]--;
         xx[1]=1;
         radom = true;
       }
    else
       { xx[0]--;
```

```
            xx[1]=0;
            radom = true;
          }
      }
     return erl3_miu1;
    } // case 7 pabaiga

    case 8:
    {
     if ( xx[1]==7)
       {
        if ( xx[0]>1)
           { xx[0]--;
             xx[1]=1;
             radom = true;
           }
        else
           { xx[0]--;
             xx[1]=0;
             radom = true;
           }
       }
     return erl3_miu2;
    } // case 8 pabaiga

   }     // switch pabaiga
 }  // erl3_n lygu 3

// ----------------------erlango MG1 4-------------------------------------------
 if (erl3_n==4)
 {
  switch(poz)
   {
    case 0:
    {
     if( (xx[1]==0)&&(xx[0]<L))
        {xx[0]++;xx[1]=1;radom=true;}
    else if (xx[0]<L)
           { xx[0]++; radom=true; }
     return lamda;
    } // case 0 pabaiga
    case 1:
    {
      if (xx[1]==1)
         {xx[1]=2;radom = true; }
       return erl3_p*100000;
    } // case 1 pabaiga
    case 2:
    {
      if (xx[1]==1)  { xx[1]=3;radom = true;}
       return (1-erl3_p)*100000;
    } // case 2 pabaiga
     case 3:
    {
      if (xx[1]==2)
         { xx[1]=4; radom = true; }
       return erl3_miu1;
    } // case 3 pabaiga
     case 4:
    {if (xx[1]==3)
         {  xx[1]=5; radom = true;}
       return erl3_miu2;
    } // case 4 pabaiga
     case 5:
    {
      if (xx[1]==4) { xx[1]=6;radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 6:
    { if (xx[1]==5){  xx[1]=7; radom = true; }
 return erl3_miu2;
    } // case 6 pabaiga
     case 7:
    {
      if (xx[1]==6) { xx[1]=8; radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 8:
    { if (xx[1]==7){ xx[1]=9; radom = true;}
 return erl3_miu2;
```

```
    } // case 6 pabaiga
    case 9:
    {if ( xx[1]==8)
       { if ( xx[0]>1){ xx[0]--; xx[1]=1; radom = true; }
        else{ xx[0]--;xx[1]=0; radom = true;  }
       }
     return erl3_miu1;
    } // case 9 pabaiga
    case 10:
    {
     if ( xx[1]==9)
       { if ( xx[0]>1){ xx[0]--; xx[1]=1;radom = true; }
        else { xx[0]--;xx[1]=0;radom = true; } }
     return erl3_miu2;
    } // case 10 pabaiga
   }     // switch pabaiga
 }  // erl3_n lygu 4
// ----------------------erlango MG1 5-----------------------------------------
  if (erl3_n==5)
 {
  switch(poz)
   {
    case 0:
    {
     if( (xx[1]==0)&&(xx[0]<L))
        {xx[0]++;xx[1]=1;radom=true;}
    else if (xx[0]<L)
            { xx[0]++; radom=true; }
     return lamda;
    } // case 0 pabaiga
    case 1:
    {
      if (xx[1]==1)
         {xx[1]=2;radom = true; }
       return erl3_p*100000;
    } // case 1 pabaiga
    case 2:
    {
      if (xx[1]==1)  { xx[1]=3;radom = true;}
       return (1-erl3_p)*100000;
    } // case 2 pabaiga
     case 3:
    {
      if (xx[1]==2)
         { xx[1]=4; radom = true; }
       return erl3_miu1;
    } // case 3 pabaiga
     case 4:
    {if (xx[1]==3)
         {  xx[1]=5; radom = true;}
       return erl3_miu2;
    } // case 4 pabaiga
     case 5:
    {
      if (xx[1]==4) { xx[1]=6;radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 6:
    { if (xx[1]==5){  xx[1]=7; radom = true; }
 return erl3_miu2;
    } // case 6 pabaiga
     case 7:
    {
      if (xx[1]==6) { xx[1]=8; radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 8:
    { if (xx[1]==7){ xx[1]=9; radom = true;}
 return erl3_miu2;
    } // case 6 pabaiga
    case 9:
    {
      if (xx[1]==8) { xx[1]=10; radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 10:
    { if (xx[1]==9){ xx[1]=11; radom = true;}
 return erl3_miu2;
    } // case 6 pabaiga
    case 11:
```

```
   {if ( xx[1]==10)
      { if ( xx[0]>1){ xx[0]--; xx[1]=1; radom = true; }
       else{ xx[0]--;xx[1]=0; radom = true;  }
      }
    return erl3_miu1;
   } // case 9 pabaiga
   case 12:
   {
    if ( xx[1]==11)
      { if ( xx[0]>1){ xx[0]--; xx[1]=1;radom = true; }
       else { xx[0]--;xx[1]=0;radom = true; } }
    return erl3_miu2;
   } // case 10 pabaiga
  }     // switch pabaiga
}  // erl3_n lygu 5
// ----------------------erlango MG1 6-------------------------------------------
 if (erl3_n==6)
{
 switch(poz)
  {
   case 0:
   {
    if( (xx[1]==0)&&(xx[0]<L))
       {xx[0]++;xx[1]=1;radom=true;}
   else if (xx[0]<L)
           { xx[0]++; radom=true; }
    return lamda;
   } // case 0 pabaiga
   case 1:
   {
     if (xx[1]==1)
        {xx[1]=2;radom = true; }
      return erl3_p*100000;
   } // case 1 pabaiga
   case 2:
   {
     if (xx[1]==1)  { xx[1]=3;radom = true;}
      return (1-erl3_p)*100000;
   } // case 2 pabaiga
    case 3:
   {
     if (xx[1]==2)
        { xx[1]=4; radom = true; }
      return erl3_miu1;
   } // case 3 pabaiga
    case 4:
   {if (xx[1]==3)
        {  xx[1]=5; radom = true;}
      return erl3_miu2;
   } // case 4 pabaiga
    case 5:
   {
     if (xx[1]==4) { xx[1]=6;radom = true; }
      return erl3_miu1;
   } // case 5 pabaiga
    case 6:
   { if (xx[1]==5){  xx[1]=7; radom = true; }
return erl3_miu2;
   } // case 6 pabaiga
    case 7:
   {
     if (xx[1]==6) { xx[1]=8; radom = true; }
      return erl3_miu1;
   } // case 5 pabaiga
    case 8:
   { if (xx[1]==7){ xx[1]=9; radom = true;}
return erl3_miu2;
   } // case 6 pabaiga
   case 9:
   {
     if (xx[1]==8) { xx[1]=10; radom = true; }
      return erl3_miu1;
   } // case 5 pabaiga
    case 10:
   { if (xx[1]==9){ xx[1]=11; radom = true;}
return erl3_miu2;
   } // case 6 pabaiga
   case 11:
   {  if (xx[1]==10) { xx[1]=12; radom = true; }  return erl3_miu1;
   } // case 5 pabaiga
```

```
    case 12:
   { if (xx[1]==11){ xx[1]=13; radom = true;}return erl3_miu2;
   } // case 6 pabaiga
   case 13:
   {if ( xx[1]==12)
      { if ( xx[0]>1){ xx[0]--; xx[1]=1; radom = true; }
       else{ xx[0]--;xx[1]=0; radom = true;  }
      }
    return erl3_miu1;
   } // case 9 pabaiga
   case 14:
   {
    if ( xx[1]==13)
      { if ( xx[0]>1){ xx[0]--; xx[1]=1;radom = true; }
       else { xx[0]--;xx[1]=0;radom = true; } }
    return erl3_miu2;
   } // case 10 pabaiga
  }     // switch pabaiga
}  // erl3_n lygu 6

 // ----------------------erlango MG1 7-----------------------------------------
 if (erl3_n==7)
{
 switch(poz)
  {
   case 0:
   {
    if( (xx[1]==0)&&(xx[0]<L))
       {xx[0]++;xx[1]=1;radom=true;}
   else if (xx[0]<L)
           { xx[0]++; radom=true; }
    return lamda;
   } // case 0 pabaiga
   case 1:
   {
     if (xx[1]==1)
        {xx[1]=2;radom = true; }
      return erl3_p*100000;
   } // case 1 pabaiga
   case 2:
   {
     if (xx[1]==1)  { xx[1]=3;radom = true;}
      return (1-erl3_p)*100000;
   } // case 2 pabaiga
    case 3:
   {
     if (xx[1]==2)
        { xx[1]=4; radom = true; }
      return erl3_miu1;
   } // case 3 pabaiga
    case 4:
   {if (xx[1]==3)
        {  xx[1]=5; radom = true;}
      return erl3_miu2;
   } // case 4 pabaiga
    case 5:
   {
     if (xx[1]==4) { xx[1]=6;radom = true; }
      return erl3_miu1;
   } // case 5 pabaiga
    case 6:
   { if (xx[1]==5){  xx[1]=7; radom = true; }
return erl3_miu2;
   } // case 6 pabaiga
    case 7:
   {
     if (xx[1]==6) { xx[1]=8; radom = true; }
      return erl3_miu1;
   } // case 5 pabaiga
    case 8:
   { if (xx[1]==7){ xx[1]=9; radom = true;}
return erl3_miu2;
   } // case 6 pabaiga
   case 9:
   {
     if (xx[1]==8) { xx[1]=10; radom = true; }
      return erl3_miu1;
   } // case 5 pabaiga
    case 10:
   { if (xx[1]==9){ xx[1]=11; radom = true;}
```

```
return erl3_miu2;
   } // case 6 pabaiga
   case 11:
   {  if (xx[1]==10) { xx[1]=12; radom = true; }  return erl3_miu1;
   } // case 5 pabaiga
    case 12:
   { if (xx[1]==11){ xx[1]=13; radom = true;}return erl3_miu2;
   } // case 6 pabaiga
   case 13:
   {  if (xx[1]==12) { xx[1]=14; radom = true; }  return erl3_miu1;
   } // case 5 pabaiga
    case 14:
   { if (xx[1]==13){ xx[1]=15; radom = true;}return erl3_miu2;
   } // case 6 pabaiga
   case 15:
   {if ( xx[1]==14)
      { if ( xx[0]>1){ xx[0]--; xx[1]=1; radom = true; }
       else{ xx[0]--;xx[1]=0; radom = true;  }
      }
    return erl3_miu1;
   } // case 9 pabaiga
   case 16:
   {
    if ( xx[1]==15)
      { if ( xx[0]>1){ xx[0]--; xx[1]=1;radom = true; }
       else { xx[0]--;xx[1]=0;radom = true; } }
    return erl3_miu2;
   } // case 10 pabaiga
  }     // switch pabaiga
}  // erl3_n lygu 7

 // ----------------------erlango MG1 8-----------------------------------------
  if (erl3_n==8)
{
 switch(poz)
  {
   case 0:
   {
    if( (xx[1]==0)&&(xx[0]<L))
       {xx[0]++;xx[1]=1;radom=true;}
   else if (xx[0]<L)
           { xx[0]++; radom=true; }
    return lamda;
   } // case 0 pabaiga
   case 1:
   {
     if (xx[1]==1)
        {xx[1]=2;radom = true; }
      return erl3_p*100000;
   } // case 1 pabaiga
   case 2:
   {
     if (xx[1]==1)  { xx[1]=3;radom = true;}
      return (1-erl3_p)*100000;
   } // case 2 pabaiga
    case 3:
   {
     if (xx[1]==2)
        { xx[1]=4; radom = true; }
      return erl3_miu1;
   } // case 3 pabaiga
    case 4:
   {if (xx[1]==3)
        {  xx[1]=5; radom = true;}
      return erl3_miu2;
   } // case 4 pabaiga
    case 5:
   {
     if (xx[1]==4) { xx[1]=6;radom = true; }
      return erl3_miu1;
   } // case 5 pabaiga
    case 6:
   { if (xx[1]==5){  xx[1]=7; radom = true; }
return erl3_miu2;
   } // case 6 pabaiga
    case 7:
   {
     if (xx[1]==6) { xx[1]=8; radom = true; }
      return erl3_miu1;
   } // case 5 pabaiga
```

```
    case 8:
   { if (xx[1]==7){ xx[1]=9; radom = true;}
return erl3_miu2;
   } // case 6 pabaiga
   case 9:
   {
     if (xx[1]==8) { xx[1]=10; radom = true; }
      return erl3_miu1;
   } // case 5 pabaiga
    case 10:
   { if (xx[1]==9){ xx[1]=11; radom = true;}
return erl3_miu2;
   } // case 6 pabaiga
   case 11:
   {  if (xx[1]==10) { xx[1]=12; radom = true; }  return erl3_miu1;
   } // case 5 pabaiga
    case 12:
   { if (xx[1]==11){ xx[1]=13; radom = true;}return erl3_miu2;
   } // case 6 pabaiga
   case 13:
   {  if (xx[1]==12) { xx[1]=14; radom = true; }  return erl3_miu1;
   } // case 5 pabaiga
    case 14:
   { if (xx[1]==13){ xx[1]=15; radom = true;}return erl3_miu2;
   } // case 6 pabaiga
   case 15:
   {  if (xx[1]==14) { xx[1]=16; radom = true; }  return erl3_miu1;
   } // case 5 pabaiga
    case 16:
   { if (xx[1]==15){ xx[1]=17; radom = true;}return erl3_miu2;
   } // case 6 pabaiga
   case 17:
   {if ( xx[1]==16)
      { if ( xx[0]>1){ xx[0]--; xx[1]=1; radom = true; }
       else{ xx[0]--;xx[1]=0; radom = true;  }
      }
    return erl3_miu1;
   } // case 9 pabaiga
   case 18:
   {
    if ( xx[1]==17)
      { if ( xx[0]>1){ xx[0]--; xx[1]=1;radom = true; }
       else { xx[0]--;xx[1]=0;radom = true; } }
    return erl3_miu2;
   } // case 10 pabaiga
  }     // switch pabaiga
}  // erl3_n lygu 8

 // ----------------------erlango MG1 9-------------------------------------------
   if (erl3_n==9)
{
 switch(poz)
  {
   case 0:
   {
    if( (xx[1]==0)&&(xx[0]<L))
       {xx[0]++;xx[1]=1;radom=true;}
   else if (xx[0]<L)
           { xx[0]++; radom=true; }
    return lamda;
   } // case 0 pabaiga
   case 1:
   {
     if (xx[1]==1)
        {xx[1]=2;radom = true; }
      return erl3_p*100000;
   } // case 1 pabaiga
   case 2:
   {
     if (xx[1]==1)  { xx[1]=3;radom = true;}
      return (1-erl3_p)*100000;
   } // case 2 pabaiga
    case 3:
   {
     if (xx[1]==2)
        { xx[1]=4; radom = true; }
      return erl3_miu1;
   } // case 3 pabaiga
    case 4:
   {if (xx[1]==3)
```

```
        {  xx[1]=5; radom = true;}
       return erl3_miu2;
    } // case 4 pabaiga
     case 5:
    {
      if (xx[1]==4) { xx[1]=6;radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 6:
    { if (xx[1]==5){  xx[1]=7; radom = true; }
return erl3_miu2;
    } // case 6 pabaiga
     case 7:
    {
      if (xx[1]==6) { xx[1]=8; radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 8:
    { if (xx[1]==7){ xx[1]=9; radom = true;}
return erl3_miu2;
    } // case 6 pabaiga
    case 9:
    {
      if (xx[1]==8) { xx[1]=10; radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 10:
    { if (xx[1]==9){ xx[1]=11; radom = true;}
return erl3_miu2;
    } // case 6 pabaiga
    case 11:
    {  if (xx[1]==10) { xx[1]=12; radom = true; }  return erl3_miu1;
    } // case 5 pabaiga
     case 12:
    { if (xx[1]==11){ xx[1]=13; radom = true;}return erl3_miu2;
    } // case 6 pabaiga
    case 13:
    {  if (xx[1]==12) { xx[1]=14; radom = true; }  return erl3_miu1;
    } // case 5 pabaiga
     case 14:
    { if (xx[1]==13){ xx[1]=15; radom = true;}return erl3_miu2;
    } // case 6 pabaiga
    case 15:
    {  if (xx[1]==14) { xx[1]=16; radom = true; }  return erl3_miu1;
    } // case 5 pabaiga
     case 16:
    { if (xx[1]==15){ xx[1]=17; radom = true;}return erl3_miu2;
    } // case 6 pabaiga
    case 17:
    {  if (xx[1]==16) { xx[1]=18; radom = true; }  return erl3_miu1;
    } // case 5 pabaiga
     case 18:
    { if (xx[1]==17){ xx[1]=19; radom = true;}return erl3_miu2;
    } // case 6 pabaiga
    case 19:
    {if ( xx[1]==18)
       { if ( xx[0]>1){ xx[0]--; xx[1]=1; radom = true; }
        else{ xx[0]--;xx[1]=0; radom = true;  }
       }
     return erl3_miu1;
    } // case 9 pabaiga
    case 20:
    {
     if ( xx[1]==19)
       { if ( xx[0]>1){ xx[0]--; xx[1]=1;radom = true; }
        else { xx[0]--;xx[1]=0;radom = true; } }
     return erl3_miu2;
    } // case 10 pabaiga
  }     // switch pabaiga
}  // erl3_n lygu 9
// ----------------------erlango MG1 10-------------------------------------------
   if (erl3_n==10)
{
 switch(poz)
  {
   case 0:
   {
    if( (xx[1]==0)&&(xx[0]<L))
       {xx[0]++;xx[1]=1;radom=true;}
   else if (xx[0]<L)
```

```
            { xx[0]++; radom=true; }
     return lamda;
    } // case 0 pabaiga
    case 1:
    {
      if (xx[1]==1)
         {xx[1]=2;radom = true; }
       return erl3_p*100000;
    } // case 1 pabaiga
    case 2:
    {
      if (xx[1]==1)  { xx[1]=3;radom = true;}
       return (1-erl3_p)*100000;
    } // case 2 pabaiga
     case 3:
    {
      if (xx[1]==2)
         { xx[1]=4; radom = true; }
       return erl3_miu1;
    } // case 3 pabaiga
     case 4:
    {if (xx[1]==3)
         {  xx[1]=5; radom = true;}
       return erl3_miu2;
    } // case 4 pabaiga
     case 5:
    {
      if (xx[1]==4) { xx[1]=6;radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 6:
    { if (xx[1]==5){  xx[1]=7; radom = true; }
return erl3_miu2;
    } // case 6 pabaiga
     case 7:
    {
      if (xx[1]==6) { xx[1]=8; radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 8:
    { if (xx[1]==7){ xx[1]=9; radom = true;}
return erl3_miu2;
    } // case 6 pabaiga
    case 9:
    {
      if (xx[1]==8) { xx[1]=10; radom = true; }
       return erl3_miu1;
    } // case 5 pabaiga
     case 10:
    { if (xx[1]==9){ xx[1]=11; radom = true;}
return erl3_miu2;
    } // case 6 pabaiga
    case 11:
    {  if (xx[1]==10) { xx[1]=12; radom = true; }  return erl3_miu1;
    } // case 5 pabaiga
     case 12:
    { if (xx[1]==11){ xx[1]=13; radom = true;}return erl3_miu2;
    } // case 6 pabaiga
    case 13:
    {  if (xx[1]==12) { xx[1]=14; radom = true; }  return erl3_miu1;
    } // case 5 pabaiga
     case 14:
    { if (xx[1]==13){ xx[1]=15; radom = true;}return erl3_miu2;
    } // case 6 pabaiga
    case 15:
    {  if (xx[1]==14) { xx[1]=16; radom = true; }  return erl3_miu1;
    } // case 5 pabaiga
     case 16:
    { if (xx[1]==15){ xx[1]=17; radom = true;}return erl3_miu2;
    } // case 6 pabaiga
    case 17:
    {  if (xx[1]==16) { xx[1]=18; radom = true; }  return erl3_miu1;
    } // case 5 pabaiga
     case 18:
    { if (xx[1]==17){ xx[1]=19; radom = true;}return erl3_miu2;
    } // case 6 pabaiga
    case 19:
    {  if (xx[1]==18) { xx[1]=20; radom = true; }  return erl3_miu1;
    } // case 5 pabaiga
     case 20:
```

```
    { if (xx[1]==19){ xx[1]=21; radom = true;}return erl3_miu2;
    } // case 6 pabaiga
    case 21:
    {if ( xx[1]==20)
       { if ( xx[0]>1){ xx[0]--; xx[1]=1; radom = true; }
        else{ xx[0]--;xx[1]=0; radom = true;  }
       }
     return erl3_miu1;
    } // case 9 pabaiga
    case 22:
    {
     if ( xx[1]==21)
       { if ( xx[0]>1){ xx[0]--; xx[1]=1;radom = true; }
        else { xx[0]--;xx[1]=0;radom = true; } }
     return erl3_miu2;
    } // case 10 pabaiga
   }     // switch pabaiga
 }  // erl3_n lygu 9
//------------------------------

} // BusenosErlangoMG  pabaiga


//---------------------------------------------------------------------------
void TForm1::KasuSk(AnsiString DF)
{
 char elem='a';
 float elem1=0;
 FILE *F;
 F = fopen(DF.c_str(),"r");
 K=0;
 IK=0;

 //nuskaitom kiek bus busenos skaitmenu
 while (elem!='=') fscanf(F, "%c", &elem);
 fscanf(F, "%c", &elem);
 while (elem!=';')
  {
   fscanf(F, "%f%c", &elem1, &elem);
   K++;
  }

 elem='a';
 elem1=0.;
 //nuskaitom kiek bus intensyvumu
 while (elem!='=') fscanf(F, "%c", &elem);
 fscanf(F, "%c", &elem);
 while (elem!=';')
  {
   fscanf(F, "%f%c", &elem1, &elem);
   IK++;
  }
 fclose(F);
}
//---------------------------------------------------------------------------
void TForm1::Statistika_S5()       //ivairi statistika sistemai S5- 1 srautas 2 aptranavimo sistemos
{
 TBus *X=TBPr;
 int i=0;
 float* A=new float[8];
 for(i=0;i<8;i++) A[i]=0;
 i=0;
  while (X)
  {
   A[0]=A[0]+X->sk[0]*r[i];
   if (X->sk[1]>0) A[1]=A[1]+(X->sk[0]-1)*r[i];
   else A[1]=A[1]+X->sk[0]*r[i];
   if (X->sk[0]==0) A[2]=A[2]+r[i];
   if (X->sk[0]==1) A[3]=A[3]+r[i];
   if (X->sk[0]==2) A[4]=A[4]+r[i];
   if (X->sk[0]==3) A[5]=A[5]+r[i];
   if (X->sk[0]==4) A[6]=A[6]+r[i];
   if (X->sk[0]==5) A[7]=A[7]+r[i];
   X=X->kitas;
   i++;
  }
 Memo1->Lines->Add("L   "+ FloatToStrF(A[0],ffFixed,5,4));
 Memo1->Lines->Add("Lq- "+ FloatToStrF(A[1],ffFixed,5,4));
 Memo1->Lines->Add("p0- "+ FloatToStrF(A[2],ffFixed,5,4));
 Memo1->Lines->Add("p1- "+ FloatToStrF(A[3],ffFixed,5,4));
```

```
 Memo1->Lines->Add("p2- "+ FloatToStrF(A[4],ffFixed,5,4));
 Memo1->Lines->Add("p3- "+ FloatToStrF(A[5],ffFixed,5,4));
 Memo1->Lines->Add("p4- "+ FloatToStrF(A[6],ffFixed,5,4));
 Memo1->Lines->Add("p5- "+ FloatToStrF(A[7],ffFixed,5,4));
}
//---------------------------------------------------------------------------
void TForm1::StatistikaKoksoGM1()
{
 TBus *X=TBPr;
 int i=0;
 float* A=new float[8];
 float sig, ro, a0,a1,a2,a3,a4,a5,La;
 for(i=0;i<8;i++) A[i]=0;
 i=0;
  while (X)
  {
   A[0]=A[0]+X->sk[1]*r[i];
   if (X->sk[2]>0) A[1]=A[1]+(X->sk[1]-1)*r[i];
   else A[1]=A[1]+X->sk[1]*r[i];
   if (X->sk[1]==0) A[2]=A[2]+r[i];
   if (X->sk[1]==1) A[3]=A[3]+r[i];
   if (X->sk[1]==2) A[4]=A[4]+r[i];
   if (X->sk[1]==3) A[5]=A[5]+r[i];
   if (X->sk[1]==4) A[6]=A[6]+r[i];
   if (X->sk[1]==5) A[7]=A[7]+r[i];
   X=X->kitas;
   i++;
  }
 ro = (1/m1)*(1/lamda);
 sig = 1- ro/A[0];
 La = sig/(1-sig);
 a0 = 1-sig;
 a1 = (1-sig)*sig;
 a2 = (1-sig)*sig*sig;
 a3 = (1-sig)*sig*sig*sig;
 a4 = (1-sig)*sig*sig*sig*sig;
 a5 = (1-sig)*sig*sig*sig*sig*sig;
//---------------------------------------------------------------------------
void TForm1::StatistikaKoksoMG1()
{
  TBus *X=TBPr;
 int i=0;
 float* A=new float[8];
 for(i=0;i<8;i++) A[i]=0;
 i=0;
  while (X)
  {
   A[0]=A[0]+X->sk[0]*r[i];
   if (X->sk[1]>0) A[1]=A[1]+(X->sk[0]-1)*r[i];
   else A[1]=A[1]+X->sk[0]*r[i];
   if (X->sk[0]==0) A[2]=A[2]+r[i];
   if (X->sk[0]==1) A[3]=A[3]+r[i];
   if (X->sk[0]==2) A[4]=A[4]+r[i];
   if (X->sk[0]==3) A[5]=A[5]+r[i];
   if (X->sk[0]==4) A[6]=A[6]+r[i];
   if (X->sk[0]==5) A[7]=A[7]+r[i];
   X=X->kitas;
   i++;
  }
}
//---------------------------------------------------------------------------
void TForm1::Momentai()
{
}
//---------------------------------------------------------------------------
void TForm1::Erlango2()
{
 m1 = 0.9094;
 m2 = 1.8589;
 c = sqrt(m2-m1*m1)/m1;
 c2 = c*c;
 asim = m3-3*m1*m2+2*m1*m1*m1/((m2-m1*m1)*sqrt(m2-m1*m1));
 erl2_n = ceil(1/c2);
 erl2_p=(1/(1+c2))*(erl2_n*c2-sqrt(erl2_n*(1+c2)-erl2_n*erl2_n*c2));
 erl2_miu = (erl2_n-erl2_p)/m1;
 Memo1->Lines->Add(IntToStr(erl2_n));
 Memo1->Lines->Add(FloatToStrF(erl2_p,ffFixed,6,4));
 Memo1->Lines->Add(FloatToStrF(erl2_miu,ffFixed,6,4));

}
```

```
void TForm1::Kokso2()
{
 m1 = 0.9094;
 m2 = 1.8589;
 c = sqrt(m2-m1*m1)/m1;
 c2 = c*c;
 asim = m3-3*m1*m2+2*m1*m1*m1/((m2-m1*m1)*sqrt(m2-m1*m1));
 Memo1->Lines->Add("Kokso2");
 kok2_miu1=2/m1;
 kok2_p=0.5/c2;
 kok2_miu2=kok2_miu1*kok2_p;
 Memo1->Lines->Add(FloatToStrF(kok2_p,ffFixed,6,4));
 Memo1->Lines->Add(FloatToStrF(kok2_miu1,ffFixed,6,4));
 Memo1->Lines->Add(FloatToStrF(kok2_miu2,ffFixed,6,4));
}

void TForm1::Kokso3()
{
 float f1,f2,f3,D;
 float mc2, mc3;
 mc2 = m2/(m1*m1);
 mc3 = m3/(m1*m2);
 if ( (4*mc2/3<=mc3)&&(mc3<=6*(mc2-1)/mc2)&&(3/2<=mc2)&&(mc2<=2)
         || ((4*mc2/3<=mc3)&&(mc2>2)))
    {
      c = sqrt(m2-m1*m1)/m1;
      c2 = c*c;
      // asim = m3-3*m1*m2+2*m1*m1*m1/((m2-m1*m1)*sqrt(m2-m1*m1));
      f1 = m1;
      f2 = m2/2;
      f3 = m3/6;
      c2 = (m2-m1*m1)/(m1*m1);
      D = (f1*f2-f3)*(f1*f2-f3)-4*(f2*f2-f1*f3)*(f1*f1-f2);
      kok3_miu2 = ((f1*f2-f3)+sqrt(D))/(2*(f2*f2-f1*f3));
      kok3_miu1 = (kok3_miu2*f1-1)/(kok3_miu2*f2-f1);
      kok3_p=(kok3_miu2*(f1*kok3_miu1-1))/kok3_miu1;
      Memo1->Lines->Add("Duomenys tenkina Kokso aproksimacijos salygas");
      Button3->Enabled = true;
      Button4->Enabled = true;
      Memo1->Lines->Add("Kokso skirstinio parametrai:");
      Memo1->Lines->Add("p = "+FloatToStrF(kok3_p,ffFixed,6,4));
      Memo1->Lines->Add("miu1 = "+FloatToStrF(kok3_miu1,ffFixed,6,4));
      Memo1->Lines->Add("miu2 = "+FloatToStrF(kok3_miu2,ffFixed,6,4));
    }
 else
    Memo1->Lines->Add("Kokso aproksimacija siem duomenim netinka");
}

void TForm1::Erlango3()
{
 float xx,yy,A,B,C;
 Button6->Enabled = true;
 Button7->Enabled = true;
 erl3_n = max( ceil(1/(c*c)),ceil((-asim+1/(c*c*c)+1/c+2*c)/(asim-c+1/c)));
 xx = m1*m3-m2*m2*(erl3_n+2)/(erl3_n+1);
 yy = m2-m1*m1*(erl3_n+1)/erl3_n;
 C = m1*xx;
 B = -(erl3_n*xx+yy*yy*(erl3_n*erl3_n+2*erl3_n)/(erl3_n+1)+(erl3_n+2)*yy*m1*m1);
 A = erl3_n*(erl3_n+2)*m1*yy;
 erl3_miu1 = 2*A/(sqrt(B*B-4*A*C)-B);
 erl3_miu2 = 2*A/(-sqrt(B*B-4*A*C)-B);
 erl3_p = (m1/erl3_n-1/erl3_miu2)/(1/erl3_miu1-1/erl3_miu2);
 Button5->Enabled = true;
 Button6->Enabled = true;
 Memo1->Lines->Add("Erlango misinio  parametrai:");
 Memo1->Lines->Add("n = "+IntToStr(erl3_n));
 Memo1->Lines->Add("p = "+FloatToStrF(erl3_p,ffFixed,6,4));
 Memo1->Lines->Add("miu1 = "+FloatToStrF(erl3_miu1,ffFixed,6,4));
 Memo1->Lines->Add("miu2 = "+FloatToStrF(erl3_miu2,ffFixed,6,4));

}
//---------------------------------------------------------------------
void TForm1::Isvedimas(int tipas)
{
 if (M>0)
 {
  BusenuRadimas(tipas);
  Apjungimas();
```

```
  IsvestiStrGrid();
  //else TabSheet3->TabVisible=false;
  StacTikRadimas();
  //IsvestiMemo();

  if (tipas==1) StatistikaKoksoMG1();
  if (tipas==2) StatistikaKoksoGM1();
  if (tipas==3) StatistikaKoksoMG1();
  if (tipas==4) StatistikaKoksoGM1();

 }

 TBusIntens *D;
 while (TPr)
  {D=TPr; TPr=TPr->kitas; delete D;}
 TBus *DD;
 while (TBPr)
  {DD=TBPr; TBPr=TBPr->kitas; delete DD;}
}
//---------------------------------------------------------------------------


void __fastcall TForm1::Button4Click(TObject *Sender)
{
 K=0;
 M=0;

 IvestiPradSalygasKoksoGM1();
 Isvedimas(2);
 PageControl1->Pages[1]->Show();
 Button10->Enabled = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
 K=0;
 M=0;

 IvestiPradSalygasKoksoMG1();

 Isvedimas(1);
 PageControl1->Pages[1]->Show();
 Button10->Enabled = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button8Click(TObject *Sender)
{
PageControl1->Pages[0]->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button9Click(TObject *Sender)
{
 PageControl1->Pages[1]->Show();
 Button10->Enabled = true;
}
//---------------------------------------------------------------------------


void __fastcall TForm1::Button1Click(TObject *Sender)
{
 FILE *FD;
 char R[80];
 int i = 1;
 float tikr;
 bool visiteig;
 visiteig = true;
 n = 0;
 FD = fopen("Duomenys/Imties duomenys.txt","r");
 while(!feof(FD))
      { fgets(R,80,FD);
        n++;
      }
 fclose(FD);
 y = new float [n+1];
 *(y+0)=7;
 FD = fopen("Duomenys/Imties duomenys.txt","r");
 for (i=1;i<=n;i++) fscanf(FD,"%f",y+i);
 fclose(FD);
```

```
 for ( i=1;i<=n;i++)
     { tikr = *(y+i);
       if (tikr<0) visiteig = false;
     }
 if (visiteig == false) ShowMessage("Duomenu faile yra neigiamu skaitmenu");
 Memo1->Lines->Add("Duomenys nuskaityti");
 m1 = 0;
 m2 = 0;
 m3 = 0;
 for ( int i=1; i<=n; i++)
     { m1 += *(y+i);
       m2 += (*(y+i))*(*(y+i));
       m3 += (*(y+i))*(*(y+i))*(*(y+i));
     }
 m1 = m1/n;
 m2 = m2/n;
 m3 = m3/n;
 c = sqrt(m2-m1*m1)/m1;
 c2 = c*c;
 asim = m3-3*m1*m2+2*m1*m1*m1/((m2-m1*m1)*sqrt(m2-m1*m1));
 Memo1->Lines->Add(FloatToStrF(m1,ffFixed,6,2));
 Memo1->Lines->Add(FloatToStrF(m2,ffFixed,6,2));
 Memo1->Lines->Add(FloatToStrF(m3,ffFixed,6,2));
 Edit1->SetFocus();
 Edit2->Text = "1,00";

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
 FILE *FD;
 char R[80];
 int i = 1;
 float tikr;
 bool visiteig;
 visiteig = true;
 n = 0;
 FD = fopen("Duomenys/Tikslus duomenys.txt","r");

 while(!feof(FD))
      { fgets(R,80,FD);
        n++;
      }
 fclose(FD);
 if (n<3)  ShowMessage("Duomenu faile truksta duomenu");
 else
 {
 FD = fopen("Duomenys/Tikslus duomenys.txt","r");
 fscanf(FD,"%f\n",&m1);
 fscanf(FD,"%f\n",&m2);
 fscanf(FD,"%f/n",&m3);
 fclose(FD);
 c = sqrt(m2-m1*m1)/m1;
 c2 = c*c;
 asim = (m3-3*m1*m2+2*m1*m1*m1)/((m2-m1*m1)*sqrt(m2-m1*m1));
 Memo1->Lines->Add("Duomenys nuskaityti");
 Memo1->Lines->Add("Pradiniai skirstinio G momentai:");
 Memo1->Lines->Add("m1 = "+FloatToStrF(m1,ffFixed,6,4));
 Memo1->Lines->Add("m2 = "+FloatToStrF(m2,ffFixed,6,4));
 Memo1->Lines->Add("m3 = "+FloatToStrF(m3,ffFixed,6,4));
 Erlango3();
 Kokso3();
 Edit1->SetFocus();
 Edit2->Text = "1,00";
 }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button5Click(TObject *Sender)
{
 K=0;
 M=0;

 IvestiPradSalygasErlangoMG1();
 Isvedimas(3);
 PageControl1->Pages[1]->Show();
 Button10->Enabled = true;
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Button6Click(TObject *Sender)
{
 K=0;
 M=0;

 IvestiPradSalygasErlangoGM1();
 Isvedimas(4);
 PageControl1->Pages[1]->Show();
 Button10->Enabled = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button12Click(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------

void TForm1::StringGridas1(int eil, int stulp, double **B)
{ StringGrid1->RowCount = eil+1;
  StringGrid1->ColCount = stulp+1;
  for(int i=0; i<eil; i++)
  { for(int j=0; j<stulp; j++)
    { StringGrid1->Cells[j+1][i+1] = FloatToStrF(B[i][j],ffGeneral,6,6);
    }
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button11Click(TObject *Sender)
{
  PageControl1->Pages[0]->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button10Click(TObject *Sender)
{
  Skaidymas();
  Kms();
  Takahashi();
  PageControl1->Pages[2]->Show();
}
//---------------------------------------------------------------------------

void TForm1::Kms()
{ bl = new int[nb+1];
  bl[0] = 1;
  for(int k=0; k<nb; k++)
  { bl[k+1] = bl[k] + ni[k];
  }
  double **E = Nuline_Matrica(nn, nb);
  int next = -1;
  for(int i=0; i<nb; i++)
  { for(int k=0; k<ni[i]; k++)
    { next = next + 1;
      E[next][i] = 1;
    }
  }
  double **Pije = Matricu_Daugyba(nn, nn, P, nn, nb, E);
  double **Phi = Nuline_Matrica(nb, nn);
  for(int m=0; m<nb; m++)
  { for(int j=0; j<ni[m]; j++)
    { Phi[m][bl[m]+j-1] = (double) 1 / ni[m];
    }
  }
  double **A = Matricu_Daugyba(nb, nn, Phi, nn, nb, Pije);
  double **AA = Transponuota(nb, nb, Matricu_Atimtis(nb, nb, A, Vienetine_Matrica(nb)));
  double **EN = Nuline_Matrica(nb,1);
  EN[nb-1][0] = 1;
  for(int j=0; j<nb; j++) AA[nb-1][j] = 1;
  double **XI = Matricu_Daugyba(nb, nb, Atvirkstine(nb, AA), nb, 1, EN);
  double **Z = Matricu_Daugyba(nn, nb, Transponuota(nb, nn, Phi), nb, 1, XI);
  ofstream fr("Rezultatai/KMS.txt");
  for(int iter=0; iter<itmax1; iter++)
  { for(int m=0; m<nb; m++)
    { double **Pmm = Matricos_Dalis(0, bl[m], bl[m+1], P);
      double **Zz = Matricos_Dalis(1, bl[m], bl[m+1], Z);
      double **Bb = Matricu_Daugyba(1, bl[m+1]-bl[m],Transponuota(bl[m+1]-bl[m], 1, Zz), bl[m+1]-bl[m],
bl[m+1]-bl[m], Pmm);
```

```
      double **Zb = Matricu_Daugyba(1, nn, Transponuota(nn, 1, Z), nn, bl[m+1]-bl[m],
Matricos_Dalis(nn, bl[m], bl[m+1], P));//Matricos_Dalis(nn, bl[m], bl[m+1], P);//Transponuota(nn, 1,
Z);
      double **b = Matricu_Atimtis(1, bl[m+1]-bl[m], Bb, Zb);
      double **x0 = Matricos_Dalis(1, bl[m], bl[m+1], Z);
      double ** Y = Gauss_Seidel(bl[m+1]-bl[m], Matricu_Atimtis(bl[m+1]-bl[m], bl[m+1]-bl[m],
Transponuota(bl[m+1]-bl[m], bl[m+1]-bl[m], Pmm), Vienetine_Matrica(ni[m])), x0, Transponuota(1, ni[m],
b));
    //  StringGridas1(bl[m+1]-bl[m], 1, Y);
      for(int j=0; j<ni[m]; j++) Z[bl[m]+j-1][0] = Y[j][0];
      double sum_Y = 0;
      for(int j=0; j<ni[m]; j++) sum_Y = sum_Y + Y[j][0];
      for(int j=0; j<ni[m]; j++) Y[j][0] = Y[j][0] / sum_Y;
    //  StringGridas1(bl[m+1]-bl[m], 1, Y);
      for(int j=0; j<ni[m]; j++) Phi[m][bl[m]+j-1] = Y[j][0];
    }
    PI = new double[nn];
    for(int i=0; i<nn; i++) PI[i] = Z[i][0];
    fr<<"Stacionarios tikimybes(KMS): iteracija "<< iter <<endl;
     for (int j = 0; j < nn; j++)
        { fr << setw(16) << setprecision(15) << PI[j]<<endl;
        }
     fr<<"-----------------"<< endl;

    double sum_pi = 0;
    for(int i=0; i<nn; i++)
    { sum_pi = sum_pi + PI[i];
    }

    fr << setw(16) << setprecision(15) << "Stacionariu tikimybiu suma: " << sum_pi <<endl;
    fr<<"-----------------"<< endl;

    double **RES =
Matricu_Daugyba(nn,nn,Matricu_Atimtis(nn,nn,Transponuota(nn,nn,P),Vienetine_Matrica(nn)),nn,1,Z);
   // StringGridas1(nn, 1, RES);
    double sum_res = 0;
    for(int i=0; i<nn; i++)
    { if (RES[i][0] < 0) sum_res = sum_res - RES[i][0];
      else sum_res = sum_res + RES[i][0];
    }
    double **A = Matricu_Daugyba(nb, nn, Phi, nn, nb, Pije);
    double **AA = Transponuota(nb, nb, Matricu_Atimtis(nb, nb, A, Vienetine_Matrica(nb)));

    double **EN = Nuline_Matrica(nb,1);
    EN[nb-1][0] = 1;
    for(int j=0; j<nb; j++) AA[nb-1][j] = 1;
    double **XI = Matricu_Daugyba(nb, nb, Atvirkstine(nb, AA), nb, 1, EN);

    for (int i = 0; i < nb; i++)
      { for (int j = 0; j < 1; j++)
        { fr << setw(12) << setprecision(9) << XI[i][j]<<" ";
        }
        fr<< endl;
      }
      fr<<"-----------------"<< endl;

    Z = Matricu_Daugyba(nn, nb, Transponuota(nb, nn, Phi), nb, 1, XI);
  }
  fr.close();
  StringGridas2(nn, 1, PI);
}
//---------------------------------------------------------------------------

void TForm1::Takahashi()
{ bl = new int[nb+1];
  bl[0] = 1;
  for(int k=0; k<nb; k++)
  { bl[k+1] = bl[k] + ni[k];
  }
  double **E = Nuline_Matrica(nn, nb);
  int next = -1;
  for(int i=0; i<nb; i++)
  { for(int k=0; k<ni[i]; k++)
    { next = next + 1;
      E[next][i] = 1;
    }
  }
  double **Pije = Matricu_Daugyba(nn, nn, P, nn, nb, E);
  double **Phi = Nuline_Matrica(nb, nn);
  for(int m=0; m<nb; m++)
```

```
  { for(int j=0; j<ni[m]; j++)
    { Phi[m][bl[m]+j-1] = (double) 1 / ni[m];
    }
  }
  double **A = Matricu_Daugyba(nb, nn, Phi, nn, nb, Pije);
  double **AA = Transponuota(nb, nb, Matricu_Atimtis(nb, nb, A, Vienetine_Matrica(nb)));
  double **EN = Nuline_Matrica(nb,1);
  EN[nb-1][0] = 1;
  for(int j=0; j<nb; j++) AA[nb-1][j] = 1;
  double **XI = Matricu_Daugyba(nb, nb, Atvirkstine(nb, AA), nb, 1, EN);
  double **Z;
  ofstream fr("Rezultatai/Takahashi.txt");
  for(int iter=0; iter<itmax1; iter++)
  { for(int m=0; m<nb; m++)
    { double **Pmm = Matricos_Dalis(0, bl[m], bl[m+1], P);
      Z = Matricu_Daugyba(nn, nb, Transponuota(nb, nn, Phi), nb, 1, XI);
      double **Zz = Matricos_Dalis(1, bl[m], bl[m+1], Z);
      double **Bb = Matricu_Daugyba(1, bl[m+1]-bl[m],Transponuota(bl[m+1]-bl[m], 1, Zz), bl[m+1]-bl[m],
bl[m+1]-bl[m], Pmm);
      double **Zb = Matricu_Daugyba(1, nn, Transponuota(nn, 1, Z), nn, bl[m+1]-bl[m],
Matricos_Dalis(nn, bl[m], bl[m+1], P));//Matricos_Dalis(nn, bl[m], bl[m+1], P);//Transponuota(nn, 1,
Z);
      double **b = Matricu_Atimtis(1, bl[m+1]-bl[m], Bb, Zb);
      double **x0 = Matricos_Dalis(1, bl[m], bl[m+1], Z);
      double ** Y = Gauss_Seidel(bl[m+1]-bl[m], Matricu_Atimtis(bl[m+1]-bl[m], bl[m+1]-bl[m],
Transponuota(bl[m+1]-bl[m], bl[m+1]-bl[m], Pmm), Vienetine_Matrica(ni[m])), x0, Transponuota(1, ni[m],
b));
      double sum_Y = 0;
      for(int j=0; j<ni[m]; j++) sum_Y = sum_Y + Y[j][0];
      for(int j=0; j<ni[m]; j++) Y[j][0] = Y[j][0] / sum_Y;
      for(int j=0; j<ni[m]; j++) Phi[m][bl[m]+j-1] = Y[j][0];
    }
    Z = Matricu_Daugyba(nn, nb, Transponuota(nb, nn, Phi), nb, 1, XI);
    PI = new double[nn];
    for(int i=0; i<nn; i++) PI[i] = Z[i][0];
    //StringGridas3(nn, 1, PI);

     fr<<"Stacionarios tikimybes(Takahashi): iteracija "<< iter <<endl;
     for (int j = 0; j < nn; j++)
        { fr << setw(16) << setprecision(15) << PI[j]<<endl;
        }
     fr<<"-----------------"<< endl;
    double sum_pi = 0;
    for(int i=0; i<nn; i++)
    { sum_pi = sum_pi + PI[i];
    }
    fr << setw(16) << setprecision(15) << "Stacionariu tikimybiu suma: " << sum_pi <<endl;
    fr<<"-----------------"<< endl;
    double **RES =
Matricu_Daugyba(nn,nn,Matricu_Atimtis(nn,nn,Transponuota(nn,nn,P),Vienetine_Matrica(nn)),nn,1,Z);
    double sum_res = 0;
    for(int i=0; i<nn; i++)
    { if (RES[i][0] < 0) sum_res = sum_res - RES[i][0];
      else sum_res = sum_res + RES[i][0];
    }
    double **A = Matricu_Daugyba(nb, nn, Phi, nn, nb, Pije);
    double **AA = Transponuota(nb, nb, Matricu_Atimtis(nb, nb, A, Vienetine_Matrica(nb)));
    double **EN = Nuline_Matrica(nb,1);
    EN[nb-1][0] = 1;
    for(int j=0; j<nb; j++) AA[nb-1][j] = 1;
    XI = Matricu_Daugyba(nb, nb, Atvirkstine(nb, AA), nb, 1, EN);

     for (int i = 0; i < nb; i++)
      { for (int j = 0; j < 1; j++)
        { fr << setw(12) << setprecision(9) << XI[i][j]<<" ";
        }
        fr<< endl;
      }
      fr<<"-----------------"<< endl;
  }
  fr.close();
  StringGridas3(nn, 1, PI);
}
//---------------------------------------------------------------------------

double** TForm1::Nuline_Matrica(int eil, int stulp)
{ double **M;
  M = new double*[eil];
  for(int i=0; i<eil; i++)
  { M[i] = new double[stulp];
```

```
  }
  for(int i=0; i<eil; i++)
  { for(int j=0; j<stulp; j++)
    { M[i][j] = 0;
    }
  }
  return M;
}
//---------------------------------------------------------------------------

double** TForm1::Vienetine_Matrica(int eil)
{ double **M;
  M = new double*[eil];
  for(int i=0; i<eil; i++)
  { M[i] = new double[eil];
  }
  for(int i=0; i<eil; i++)
  { for(int j=0; j<eil; j++)
    { if (i==j) M[i][j] = 1;
      else M[i][j] = 0;
    }
  }
  return M;
}
//---------------------------------------------------------------------------

double** TForm1::Matricu_Daugyba(int eil_a, int stulp_a, double **A, int eil_b,
                                 int stulp_b, double **B)
{ double **AB;
  AB = new double*[eil_a];
  for(int i=0; i<eil_a; i++)
  { AB[i] = new double[stulp_b];
  }
  double suma;
  for(int k=0; k<stulp_b; k++)
  { for(int i=0; i<eil_a; i++)
    { suma = 0;
      for(int j=0; j<stulp_a; j++)
      { suma = suma + A[i][j] * B[j][k];
      }
      AB[i][k] = suma;
    }
  }
  return AB;
}
//---------------------------------------------------------------------------

double** TForm1::Matricu_Atimtis(int eil, int stulp, double **A, double **B)
{ double **AB;
  AB = new double*[eil];
  for(int i=0; i<eil; i++)
  { AB[i] = new double[stulp];
  }
  for(int i=0; i<eil; i++)
  { for(int j=0; j<stulp; j++)
    { AB[i][j] = A[i][j] - B[i][j];
    }
  }
  return AB;
}
//---------------------------------------------------------------------------

double** TForm1::Matricu_Sudetis(int eil, int stulp, double **A, double **B)
{ double **AB;
  AB = new double*[eil];
  for(int i=0; i<eil; i++)
  { AB[i] = new double[stulp];
  }
  for(int i=0; i<eil; i++)
  { for(int j=0; j<stulp; j++)
    { AB[i][j] = A[i][j] + B[i][j];
    }
  }
  return AB;
}
//---------------------------------------------------------------------------

double** TForm1::Transponuota(int eil, int stulp, double **A)
{ double **B;
  if (eil==stulp)
```

```
  { B = new double*[eil];
    for(int i=0; i<eil; i++)
    { B[i] = new double[stulp];
    }
    for(int i=0; i<eil; i++)
    { for(int j=0; j<stulp; j++)
      { B[i][j] = A[j][i];
      }
    }
  }
  else
  { B = new double*[stulp];
    for(int i=0; i<stulp; i++)
    { B[i] = new double[eil];
    }
    for(int i=0; i<stulp; i++)
    { for(int j=0; j<eil; j++)
      { B[i][j] = A[j][i];
      }
    }
  }
  return B;
}
//---------------------------------------------------------------------------

double** TForm1::Atvirkstine(int eile, double **A)
{ double **AAA;
  AAA = new double*[eile];
  for(int i=0; i<eile; i++)
  { AAA[i] = new double[eile];
  }
   for(int i=0; i<eile; i++)
    { for(int j=0; j<eile; j++)
      { AAA[i][j] = A[i][j];
      }
    }
  double **X;
  X = new double*[eile];
  for(int i=0; i<eile; i++)
  { X[i] = new double[eile];
  }
  double **B;
  B = new double*[eile];
  for(int i=0; i<eile; i++)
  { B[i] = new double[eile];
  }
  //int n = eile;
  int k,l,i,j;
  double z,s;
  for(int i = 0; i<eile;i++)
   for (int j = 0; j<eile;j++)
    if (i == j) B[i][j] = 1;
    else  B[i][j] = 0;
  for (k = 0; k < eile-1; k++)
  { l = k;
    for (i = k; i < eile; i++)
     if (abs(AAA[l][k]) < abs(AAA[i][k])) l = i;
     if (k != l)
     { for(j = k; j < eile; j++)
       { z = AAA[k][j]; AAA[k][j] = AAA[l][j];AAA[l][j] = z;   }
       for(j = 0; j < eile; j++)
       { z = B[k][j]; B[k][j] = B[l][j]; B[l][j] = z; }
     }
    for (i = k+1; i<eile;i++)
    { s = AAA[i][k] / AAA[k][k];
      AAA[i][k] = 0;
      for (j=k+1; j<eile; j++)
       AAA[i][j] = AAA[i][j]-s*AAA[k][j];
      for (j=0; j<eile; j++)
       B[i][j] = B[i][j] - s * B[k][j];
    }
  }
  for (l = 0; l<eile; l++)
  { X[eile-1][l] = B[eile-1][l] / AAA[eile-1][eile-1];
    for (i = eile-1; i>=0; i--)
    { s=0;
      for (j=i+1; j<eile; j++)
       s =  s + AAA[i][j] * X[j][l];
      X[i][l] = (B[i][l] - s) / AAA[i][i];
    }
```

```
  }
  return X;
}
//---------------------------------------------------------------------------

double** TForm1::Matricos_Dalis(int rusis, int nuo, int iki, double **A)
{ double **B;
  if (rusis == 0)
  { int eile = iki - nuo;
    B = new double*[eile];
    for(int i=0; i<eile; i++)
    { B[i] = new double[eile];
    }
    for(int i=0; i<eile; i++)
    { for(int j=0; j<eile; j++)
      { B[i][j] = A[i+nuo-1][j+nuo-1];
      }
    }
  }
  if (rusis == 1)
  { int eile = iki - nuo;
    B = new double*[eile];
    for(int i=0; i<eile; i++)
    { B[i] = new double[1];
    }
    for(int i=0; i<eile; i++)
    { B[i][0] = A[i+nuo-1][0];
    }
  }
  if (rusis == nn)
  { int stulp = iki - nuo;
    B = new double*[rusis];
    for(int i=0; i<rusis; i++)
    { B[i] = new double[stulp];
    }
    for(int i=0; i<rusis; i++)
    { for(int j=0; j<stulp; j++)
      { B[i][j] = A[i][j+nuo-1];
      }
    }
  }
  return B;
}
//---------------------------------------------------------------------------

double** TForm1::Gauss_Seidel(int m, double **A, double **X0, double ** B)
{ double **Soln;
  double **L = Nuline_Matrica(m, m);
  double **U = Nuline_Matrica(m, m);
  double **D = Nuline_Matrica(m, m);
  for(int i=0; i<m; i++)
  { for(int j=0; j<m; j++)
    { if (i == j) D[i][j] = A[i][j];
    }
  }
  for(int i=0; i<m; i++)
  { for(int j=0; j<m; j++)
    { if (i < j) U[i][j] = -A[i][j];
      if (i > j) L[i][j] = -A[i][j];
    }
  }
  double **BB = Matricu_Daugyba(m, m, Atvirkstine(m, Matricu_Atimtis(m, m, D, L)), m, m, U);
  for(int iter=0; iter<itmax2; iter++)
  { Soln = Matricu_Sudetis(m, 1, Matricu_Daugyba(m, m, BB, m, 1, X0), Matricu_Daugyba(m, m,
Atvirkstine(m, Matricu_Atimtis(m, m, D, L)), m, 1, B));
    for(int i=0; i<m; i++)
    { X0[i][0] = Soln[i][0];
    }
  }
  return Soln;
}

//---------------------------------------------------------------------------

void TForm1::StringGridas2(int eil, int stulp, double *B)
{ StringGrid2->RowCount = eil+1;
  StringGrid2->ColCount = stulp+1;
  StringGrid2->Cells[1][0]="Tikimybės";
  for(int i=0; i<eil; i++)
  { for(int j=0; j<stulp; j++)
```

```
    { StringGrid2->Cells[0][i+1]="Bûsena"+IntToStr(i+1);
      StringGrid2->Cells[j+1][i+1] = FloatToStrF(B[i],ffFixed,15,15);
    }
  }
}
//---------------------------------------------------------------------------

void TForm1::StringGridas3(int eil, int stulp, double *B)
{ StringGrid3->RowCount = eil+1;
  StringGrid3->ColCount = stulp+1;
  StringGrid3->Cells[1][0]="Tikimybës";
  for(int i=0; i<eil; i++)
  { for(int j=0; j<stulp; j++)
    { StringGrid3->Cells[0][i+1]="Bûsena"+IntToStr(i+1);
      StringGrid3->Cells[j+1][i+1] = FloatToStrF(B[i],ffFixed,15,15);
    }
  }
}
//---------------------------------------------------------------------------

void TForm1::Skaidymas()
{ double **PP;
  PP = new double*[nn];
  for(int i=0; i<nn; i++)
  { PP[i] = new double[nn];
  }
  for(int i=0; i<nn; i++)
  { for(int j=0; j<nn; j++)
    { PP[i][j] = P[i][j];
    }
  }
  int bl_kiekis = 0;
  int *bl_dydziai = new int[nn/2];
  int bl_dydis = 2;
  int bl_pradzia = 0;
  int bl_pabaiga = 2;
  bool blokai = true;
  bool eilutes, stulpeliai;
  int eiluteje, stulpelyje;

  while (blokai)
  { eilutes = true;
    stulpeliai = true;
    for(int i=bl_pradzia; i<bl_pabaiga; i++)
    { eiluteje = 0;
      for(int j=bl_pradzia; j<bl_pabaiga; j++)
      { if (PP[i][j] > 0) eiluteje = 1;
      }
      if (eiluteje == 0)
      { eilutes = false;
        break;
      }

    }
    for(int j=bl_pradzia; j<bl_pabaiga; j++)
    { stulpelyje = 0;
      for(int i=bl_pradzia; i<bl_pabaiga; i++)
      { if (PP[i][j] > 0) stulpelyje = 1;
      }
      if (stulpelyje == 0)
      { stulpeliai = false;
        break;
      }

    }
  if (eilutes && stulpeliai)
  { bl_dydziai[bl_kiekis] = bl_dydis;
    bl_kiekis++;
    bl_pradzia = bl_pradzia + bl_dydis;
    bl_dydis = 2;
    bl_pabaiga = bl_pabaiga + bl_dydis;
    if (bl_pradzia >= nn) break;
    if ((nn - bl_pradzia) == 1)
    { bl_dydziai[bl_kiekis-1] = bl_dydziai[bl_kiekis-1] + bl_dydis - 1;
      blokai = false;
    }
  }
  else
  { bl_dydis++;
    bl_pabaiga = bl_pradzia + bl_dydis;
```

```
    if (bl_pabaiga > nn)
    { bl_dydziai[bl_kiekis-1] = bl_dydziai[bl_kiekis-1] + bl_dydis - 1;
      //bl_kiekis++;
      blokai = false;
    }
  }
  }

  nb = bl_kiekis;
  ni = new int[nb];
  for(int k=0; k<nb; k++)
  { ni[k] = bl_dydziai[k];
  }
  ofstream fr("Rezultatai/Skaidymas.txt"/*, ios::app*/);
  fr << "Matricos eilë: " << nn <<endl;
  fr << "Blokø kiekis: " << nb <<endl;
  fr << "Blokø dydþiai: ";
  for(int k=0; k<nb; k++)
  { fr << ni[k] << " ";
  }
  fr << endl;
  fr << "-----------------" <<endl;
  int nuo = 0;
  int iki = ni[0];
  for(int k=0; k<nb; k++)
  { for(int i=nuo; i<iki; i++)
    { for(int j=nuo; j<iki; j++)
      { fr << setw(6) << setprecision(4) << PP[i][j]<<" ";

      }
      fr << endl;

    }
    nuo = nuo + ni[k];
    iki = iki + ni[k+1];
    fr << "-----------------" <<endl;
  }
  fr.close();

}
//---------------------------------------------------------------------------
```